1987

FRIC: an expert system to recognize fricatives

Karen A. Atkinson

Rochester Institute of Technology
School of Computer Science and Technology

FRIC
An Expert System to Recognize Fricatives

Karen A. Atkinson

A thesis, submitted to
The Faculty of the school of Computer Science and Technology,
in partial fulfillment of the requirement for the degree of
Master of Science in Computer Science

Approved by:

John A. Biles

Dr. Walter Wolf

Dr. James Hillenbrand

FRIC
An Expert System to Recognize Fricatives

Karen A. Atkinson

Title of Thesis :

FRIC
An Expert System to Recognize Fricatives

I, Karen Atkinson _____ hereby grant permission to the Wallace Memorial Library, of R.I.T., to reproduce my thesis in whole or in part. Any reproduction will not be used for commercial use or profit.

Date 5/7/87

Abstract

In building a speaker independent continuous speech understanding system, a method is needed to translate the speech signal to a phoneme string. Current recognition attempts based on statistical methods are not as accurate as desired and require vast numbers of templates for comparison. On the other hand, human experts can transcribe phoneme strings from spectrograms with a high degree of accuracy, without needing to match unidentified segments against identified templates. This paper discusses the development of Fric, an expert system for identifying fricative phonemes, intended to function as part of a phoneme identification knowledge source. Fric attempts to mimic the techniques used by human transcribers in identifying phonemes. RuleMaster, an expert system building tool developed by Radian Corporation, was used to create this system. Fric uses both forward and backward chaining as control strategies and includes an explanation system for both debugging and explanatory purposes.

Table of Contents

1. Introduction 1

2. AI Background 3

3. Speech Background 21

4. Fric 34

5. Results 47

6. Conclusions 52

Bibliography 54

Appendix 1 - Preliminary System 57

Appendix 2 - Second *Place* module 62

Appendix 3 - Final Version of Interactive System 65

Appendix 4 - Explanation Problem 73

Appendix 5 - Final System 78

Appendix 6 - C Routines 86

List of Figures

Figure A - The Anatomy of an Expert System 6

Figure B - RuleMaster Main Menu 15

Figure C - Sysed Menu 16

Figure D - Sysed with Pulldown Menu 18

Figure E - Example Table 19

Figure F - Rule Corresponding to Example Table 20

Figure G - Cross Section of Head 22

Figure H - Spectrogram of "zwiebeck" 25

Figure I - Spectrogram of "six examples" 26

Figure J - System Architecture 35

Figure K - Speech Tool Display 36

Figure L - Phoneme Builder 37

Figure M - Fricatives 39

Figure N - Test Phrases 48

1. Introduction

This paper describes Fric, a knowledge based component of a speaker independent continuous speech understanding system, which is under development at the R.I.T. Research Corporation. Spoken English is made up of about forty basic sound units, called phonemes, which can be classified according to the articulatory gestures involved in creating the sound. Fric recognizes fricatives, the sounds that are the result of noisy source energy.

The method used by Fric is feature extraction, a data interpretation technique, guided by methods human experts use when determining the sounds present in a visual representation of a speech signal. These methods were learned by researching visual representations of speech and watching experts attempt to discern phrases from these representations.

The second chapter of this thesis describes the artificial intelligence background that leads to the field of expert systems. Then expert systems are defined and the process of knowledge acquisition is discussed. A short introduction to expert system tools then precedes a description of RuleMaster, a rule-based expert tool from Radian Corporation.

A basic understanding of speech science, necessary to follow the problem description and solution direction, is found in chapter three. The method of speech production is explained along with the categorization of the basic sounds in a language. Spectrograms, the visual representation of speech, are discussed along with methodologies for reading them. Then a short review of existing speech recognition systems is made.

Chapter four describes the speech understanding system that is being developed and how Fric is expected to work as a module of this system. Knowledge engineering, the process of gathering the information necessary to implement the system, is detailed, and the development of the system is described.

The testing procedure and results are described in chapter five. Hypotheses on the problems found are presented. Comments are also included on the functionality of RuleMaster and the role of knowledge engineer.

The final chapter states the conclusions drawn from this research and the possibilities for future extensions of this thesis.

The appendices to this thesis document the development of the system. Appendix 1 contains the code for the first version of Fric. A refined version of the place module of Fric is found in appendix 2. The final version of the completely interactive system is in appendix 3. Documentation on a problem found in RuleMaster's explanation system is included in appendix 4. The final version of Fric, which is still somewhat interactive, but does answer some of its own questions, is in appendix 5. The final appendix contains the code for the C routines that are used to answer Fric's non-interactive questions.

2. Artificial Intelligence

The Artificial Intelligence community does not agree on an exact definition of intelligence, but it does agree that it includes the ability to reason and to acquire and apply knowledge as well as the abilities to perceive and manipulate physical objects. These abilities require a variety of methods for representing and processing information [45].

Artificial Intelligence seeks to process knowledge as opposed to information. Traditional computing methods are concerned with calculation and storage of data, but the emphasis in AI has shifted to the use of symbolic representations of knowledge [19]. In order to use this knowledge, reasoning techniques must be found. AI researchers attempt to formulate these knowledge representation and reasoning techniques.

Prior to the late 1960's, problem solving research was centered on finding broad general laws of thinking [21]. Typical of this research was the General Problem Solver (GPS) in which an attempt was made to find general problem solving strategies that could be applied to a wide range of domains.

GPS represented task environments as data structures consisting of objects and operators, similar to a state space representation. A problem solving technique, means-ends analysis, was introduced. This technique assumed that differences between a current world and a desired one could be classified into types and that operators could be classified by the types of differences they might reduce. Goals and subgoals could then be developed by finding an operation or series of operations that would reduce differences, resulting in a world that was closer to the desired goal. A heuristically guided search through the possible operations helped to avoid paths leading back to the present world and worlds where the new differences were harder to reduce than the old ones.

The initial success with logic problems encouraged the creators to extend the program. New ways were employed to describe objects, represent operators and deal with unordered sets of symbols. As GPS grew to handle new problems, it became increasingly inept at tasks it had previously accomplished. By 1969, its creators recommended it be put to rest [20].

This search for general rules of problem solving was not successful, and the hopes that solving artificial problems, such as logic problems and building structures in a world of blocks, would provide insight into general problem solving strategies were not realized. It became clear that as problems become more complex it is necessary to have domain specific knowledge for effective decision making [14].

It is believed that at our current level of computational abilities, general purpose problem solvers are not capable of solving complex problems [25]. More recently, research has been directed towards finding solutions within a restricted domain, where a wealth of knowledge about the subject area can be collected. Such systems are called knowledge based or expert systems.

2.1. Expert Systems

An *expert system* is defined as a computer program that has the knowledge and capability to operate at an expert's level in a limited domain [20]. This performance is made possible by access to a knowledge base and the reasoning ability to find a solution to the problem within the usual resource constraints. The trend toward attempting to solve real world problems with expert systems has dictated a need for developing techniques that allow building large stores of domain specific knowledge [14].

Present development efforts are directed towards creating systems that receive their power from the knowledge embedded within them. This domain dependent approach is justified by the fact that knowledge is necessary to make decisions. Human experts are people who function best in limited domains where they have a wealth of knowledge. Knowledge based systems guard against the loss of the knowledge and also facilitate research in the domain. Information known only to a limited number of people can be lost when the knowledgeable people are no longer accessible. If the knowledge is entered into a knowledge base, the accessibility of the expert is no longer a crucial concern. Because software is easily distributed, knowledge can be shared more readily. With this dispersal of knowledge, duplicate research efforts could be prevented and research can be stimulated.

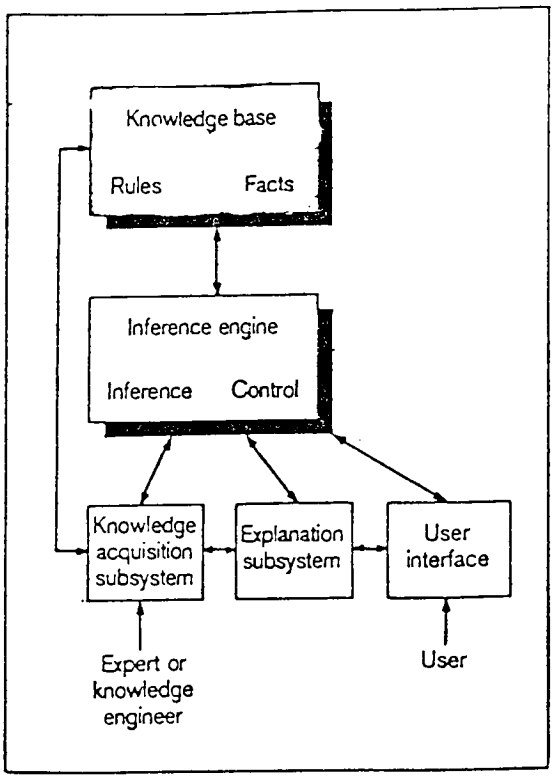
Some of the knowledge stored in an expert system is factual, and some is in the form of heuristics. Experts use heuristics, or rules of thumb, to help them decide on an approach to problem solving. Heuristics are usually acquired through years of experience working in a specific domain and therefore are not easily identified, as the expert is usually unaware of them. Experts rely on their deep understanding of the domain when heuristics fail, as this knowledge provides the raw material for developing new heuristics [8]. Concern with the collection of knowledge has led to the creation of the role of the *knowledge engineer* whose task it is to acquire from an expert, heuristic rules as well as the factual knowledge these rules will manipulate. Both types of knowledge must be successfully captured to create an effective expert system.

2.1.1. Anatomy of an Expert System

To start the process of building an expert system, one must first identify the problem domain that will be studied and locate an expert in the domain. Cooperation of a local expert is the most beneficial situation. Once this is done, information about the problem must be accumulated and analyzed to develop a knowledge representation and an inference mechanism appropriate to the goal of the program and the needs of the users. This is the task of the knowledge engineer. Next a prototype must be implemented and tested to decide the feasibility of a large scale development effort. The three main components of an expert system are the knowledge base, the inference engine and the subsystems comprising the interactive interface (Figure A) [23].

The knowledge base includes two kinds of knowledge: the factual content about the domain area and the heuristic knowledge which elucidates the problem solving strategies that experts acquire with experience. This knowledge is comprised of basic concepts in the domain and their relationships, as well as procedures to facilitate the use of these concepts in problem solving.

While learning about the domain the knowledge engineer must consider the choice of knowledge representation techniques, which could be logic, rules, a network, procedures, or



Anatomy of an Expert System

Figure A

frames and scripts among others, as well as the choice of a control strategy to guide the use of the knowledge.

Logic, one of the first knowledge representation schemes used in AI, consists of two components: axioms, which state what is known, and inference rules, which dictate what can be concluded from the asserted axioms. Logic presents a method of clearly expressing facts and their relationships in a knowledge base. Representation of heuristic knowledge, the methods which use the facts stored about the domain, is difficult using a logic representation.

Rules, also called productions, can be used to draw conclusions from existing facts. These facts can be in the form of object-attribute-value tuples or simply attribute-value pairs.

Rules are generally in the form of IF-THEN statements. Production systems are often used in areas where the knowledge consists of isolated facts, and there is no unifying theme, such as medical diagnosis.

Networks store information about objects in nodes and use the arcs connecting the nodes to represent relationships among objects. When descriptions of the relations are included in the system, it is said to be a semantic network. These structures are often used to help in parsing tasks and are extremely flexible as new nodes and arcs can be defined as needed. Networks do have the drawback that processing can be made difficult by the complex nature of the structure.

Problem solving often requires knowing how to use implicit knowledge as well as declared facts. Procedural knowledge can supply the method of extracting this implicit knowledge, such as the relationship between two nodes that are connected through an intermediary node. A procedure can be used to represent a fact by containing a sequence of instructions which, after execution, arrive at a result consistent with the fact. These instructions are often stated in the form of rules.

Representing knowledge about objects and events that are typically present in a specific situation can be achieved by frames and scripts. The frame represents a specific object in terms of its components, which may themselves be frames. Each frame has slots that store values for all the information associated with the object. Scripts contain a typical sequence of events involving frames and can be represented as a sequence of rules [16].

The inference engine guides the use of the knowledge base. The inference scheme should be simple, because the power of an expert system is in its knowledge, not in its reasoning techniques. A complex scheme can be difficult to follow, whereas a simple inference engine ensures that lines of reasoning can be readily explained [19]. This not only makes debugging easier, but also makes the system more credible to the user. Control strategies shape and restrict the representation of the procedural knowledge in the system [23]. Two inference schemes that can be used are backward and forward chaining.

Backward chaining is a goal directed control mechanism that can be used when the number of outcomes is limited. The inference engine starts by selecting a goal and then working backwards trying to validate subgoals that will eventually prove the goal valid or invalid.

Forward chaining, a data driven approach, is used when the number of possible outcomes is large. With this method the premises of rules are examined and when found to be true, the indicated conclusions are asserted as known facts. This process is then repeated until a goal state is reached [23].

As part of the interactive interface, systems may contain an explanation system, intelligent editors to aid in knowledge acquisition, and debugging and trace facilities. These utilities are present to facilitate development and use of the expert system.

Intelligent editors can provide a user interface that will facilitate knowledge acquisition. An editor knowledgeable about the structure and grammar of the knowledge base can help the user avoid typographical and semantic errors when adding or changing knowledge. In addition, an editor may also be able to discern whether newly acquired knowledge will introduce any inconsistencies into the knowledge base and so inform the user [25]. Other functions of editors include maintaining bookkeeping information so it is possible, for instance, to find the author of a particular rule.

The explanation facility should be able to explain why it needs the information it requests and how conclusions are reached. Explanations should be informational and appropriate to the level of the problem solving technique. Explanation facilities are needed in expert systems if only to aid debugging. They can help identify faulty or misused facts and rules as well as clarify the steps that led to a bad decision. Some systems offer debugging and trace facilities that allow the user to stop and examine the program during execution. Besides the debugging use, the explanation system also provides the user with a mechanism to aid in understanding the expert system's reasoning process.

2.2. Knowledge Acquisition

The primary problem facing the knowledge engineer is that of knowledge acquisition. Problem solving ability must be transferred from the human expert to the expert system, by acquiring the necessary knowledge and storing a suitable representation of it in the expert system.

After the problem domain has been defined, the basic concepts involved must be identified. Various sources of knowledge that can be used for this include textbooks, reports, databases, case studies, empirical data and personal experience [43]. These sources should be explored initially by the knowledge engineer since this type of knowledge is usually attainable without requiring the services of an expert. Gaining this level of understanding will later help the knowledge engineer converse with the expert without needing definitions of the basic terminology of the domain.

Acquiring heuristic knowledge does require an expert because this type of knowledge is rarely documented. Several techniques have been employed to extract knowledge from an expert, because a method used successfully in one domain may not function elsewhere due to the domain specific nature of expertise.

Sometimes a single method of knowledge acquisition is used while other times methods are combined. One technique requires the knowledge engineer to watch the expert solve real problems in the working environment. If the expert system is to be a control system that will interface with a number of subsystems, it is beneficial to have the knowledge engineer spend time observing the environment and noting the responses of the system controller.

Sometimes discussing with the expert the kinds of data, knowledge and procedures needed to solve specific problems is a reasonable approach. For example, when designing an expert system to approve or deny loan applications for example, time questioning a loan officer about the procedures he or she uses is well spent.

Another approach is to have the expert describe a "prototypical problem" for each category of answer in the domain. Questions that are used to this end are often in the form of "When would you suggest this particular response as the appropriate action? ".

Problem analysis, the method most frequently used in this project, consists of presenting the expert with problems to solve, while asking questions intended to explore the expert's reasoning process. The knowledge engineer cannot rely upon the expert to adequately describe the reasoning process without prompting, as the expert is usually not aware of all the individual steps in his reasoning process.

Simply requesting the rules an expert uses in problem solving is often not an effective method of knowledge acquisition. The abilities and knowledge comprising expertise often make it difficult the expert from being able to describe the knowledge used in problem solving [43]. Experts often explain their activities in general terms that are too broad for effective machine analysis, as often the expert is not aware of the individual steps in the reasoning process or may claim certain knowledge is intuitive when actually it is the result of a complex reasoning process based on vast experience [44].

2.3. Expert System Development Tools

Fortunately for the developers of expert systems, there are now tools that aid in the development process. These range in helpfulness from minimal, such as general purpose programming languages including FORTRAN and Pascal, to more helpful symbol manipulation languages such as LISP and Prolog, all the way up to knowledge engineering tools that provide an entire shell for the expert system.

Languages in general provide a mechanism for expressing the contents of the expert system, but do not create any of the utilities that will be needed, leaving this to the system creator. Therefore, the system creators must decide how to implement the user interface, which can be a very demanding project. The control mechanism used to draw inferences is included in some symbol manipulation languages, such as Prolog, but in other cases this also must be implemented.

Some of the knowledge engineering tools were derived by stripping domain specific knowledge from a functional expert system leaving a framework for building systems in other domains. These tools provide the user interface as well as the structure for implementing a

knowledge base and possibly supply a control mechanism. Examples of such tools are KAS (Knowledge Acquisition System) which was derived from Prospector, a system that guided mining operations, and Emycin (Essential Mycin), which is the shell of Mycin, a medical diagnosis program [43]. These tools speed up development time, but often lack generality and flexibility.

Other tools include EXPERT, a programming system that is designed for building expert systems that are basically classification problems. It has been used primarily for medical diagnosis problems. Knowledge is represented in hypothesis, findings and decision rules with confidence factors (a method of describing the uncertainty of a judgement) and forward chaining is used as the inference method. EXPERT does include user-interface utilities that allow the user greater ease in creating and changing the system [25].

OPS5 is another rule-based programming language that uses a data-driven inference approach. When the antecedent of a rule is satisfied, the consequent action is taken. This control, along with a method for choosing among rules with satisfied antecedents, comprises the inference engine. The form of the data structures is supplied as objects with associated attribute - value pairs, but there is no easy interface for the user and no explanation system is supplied [25].

ROSIE is a general purpose rule-based programming system that offers English-like syntax. This syntax eases the problems in creating and manipulating the knowledge base. The structure of the knowledge is in the form of object attribute - value tuples and rules. Besides backward chaining, the inference mechanisms included in ROSIE are state-driven, where the state of the system causes a rule to fire, and change-driven, where a change to the knowledge base causes a rule to fire. An explanation system and a user-friendly interface are contained within the system.

Tools can impose an undesired structure on the knowledge base or can force the use of an inappropriate inference scheme. When choosing a tool one must be aware of the needs of the expert system and match those carefully to the advantages offered by particular tools.

2.3.1. RuleMaster

RuleMaster, a software tool for building expert systems in a modular fashion, was developed by Stephen Muggleton of Intelligent Terminals Limited and Charles Riese of Radian Corporation under the direction of Donald Michie, also of ITL [39].

The RuleMaster expert system building package contains two principle components: Radial, an interpreted language for expressing and executing rules, and RuleMaker, which induces Radial rules from example tables. Figure E (page 18) shows one of these example tables. It consists of thirteen examples each containing two conditions (place and voicing), a resulting action and the next state to be entered.

RuleMaker removes the duty of rule generation from the shoulders of the system builder because, given enough specific examples, it can produce a rule to cover the situation [40]. These rules are stated in terms of a decision tree generated by inductive techniques founded on Quinlan's ID3 (Interactive Dichotomizer 3) algorithm [24,38]. This algorithm requires knowledge of all possible values of the conditions or attributes involved in classification. A set of examples including the values of the attributes and the classification illustrated must be supplied to provide the raw data for building a decision tree. Although a number of trees could be generated from an example table, RuleMaker employs a heuristic to create the smallest decision tree possible. The CLS (Concept Learning System) algorithm acts as a subroutine to the process, by attempting to find the condition which is the most discriminatory and using this condition to partition the data. This process is then repeated on the subsets of data until the subset contains only one class of data.

It is important that the number of examples provided be sufficient to form a generalization. If this is not the situation, an error message will be generated indicating the decision tree could not be built. If conflicting examples are fed to the rule inducer, another error message will indicate the problem along with the example numbers of the pieces of conflicting data. For example, if two identical sets of conditions were entered with different resulting actions, the rule inducer would refuse to create the rule and would point out the conflict to

the user. When an unnecessary condition is included in the example table, RuleMaker excludes this condition from the generated decision tree. For example, if three conditions were used to determine the age of a person, and they were the date of birth, the sex of the person, and the current date, the rule inducer would generate a rule using only the two relevant conditions: date of birth and current date. The condition concerning the sex of the person would be considered irrelevant and would be omitted. Because of this inductive approach to rule generation, modifications of the expert system are easily implemented by adding, changing or deleting examples. By the use of the rule inducer, one can write an expert system without writing any Radial code. An induction file, which contains actions, conditions, examples, and possibly some code, supplies the raw material for a Radial file. RuleMaker converts these induction files into Radial code during the induction phase.

Other features of RuleMaster aid in expert system creation. Each module is provided with the ability to feed information to the explanation facility. This facility allows the user of the system to ask why certain information is being requested or why a conclusion was reached. The explanation supplied is dependent on the system structure and creator supplied *intent* statements. The suppression of certain information can be achieved if the system designer feels the information will confuse the user rather than explain the actions of the system.

The Radial interpreter can interface with external routines written in a variety of programming languages and with external information sources other than programs such as databases, instruments and other computers. This provides a powerful interface to existing machinery and programs that may already be in use.

A RuleMaster expert system consists of a number of Radial modules, each of which is a transition network of states. Each state is essentially a decision tree, which controls transitions to other states and possibly routines written in other languages. The high level system hierarchy is stored in a framework file, which shows the control between modules and enumerates standard subroutines and text files.

RuleMaster is a menu-driven system (figure B) whose utilities may also be used directly from the operating system. It provides a duplication of UNIX file manipulation commands, the vi editor, and intelligent editors (sysed and inded) designed to implement framework and induction files.

Sysed allows the user to manage the hierarchical structure of the system, as well as access other utilities (figures C and D). Figure C shows a framework file for a system that has three top level modules : fric, total_energy and bands. The amount of indentation before a module name indicates the hierarchical level of the module. The modules is_weak and is_strong are children of the module place, which is itself a child of the module fric. Using top down management, modules are entered into the system structure and the corresponding files are created with the proper module name (the pathname to the appropriate position in the system). While in sysed, one can edit files using standard editors or with inded, a knowledge based induction file editor. Figure D shows the pulldown menu in sysed with the option to edit a file chosen. The type of editor to be used is determined by the value of the system defaults (see choice D in Figure B).

Inded can be accessed from the operating system, from sysed, or from the edit menu of RuleMaster. When editing an induction file, inded checks syntax as information is entered. If a violation of syntactic rules is encountered, code is refused and an error message is generated. When changes made to one section of an induction file should cause corresponding changes in another section, inded controls the interaction by making the necessary code implementations. The rule inducer that resides in inded allows the user to check rules after building an example table (figures E and F) and to view the induced Radial code. Figure E shows an example table for determining what sound is present and Figure F shows the rule induced from this table.

An extremely attractive feature of RuleMaster is the ability to generate C source code from the Radial code run by the interpreter. This source code is designed to be portable and therefore easily distributed. This feature of the tool allows development to take place on a

RuleMaster Main Menu

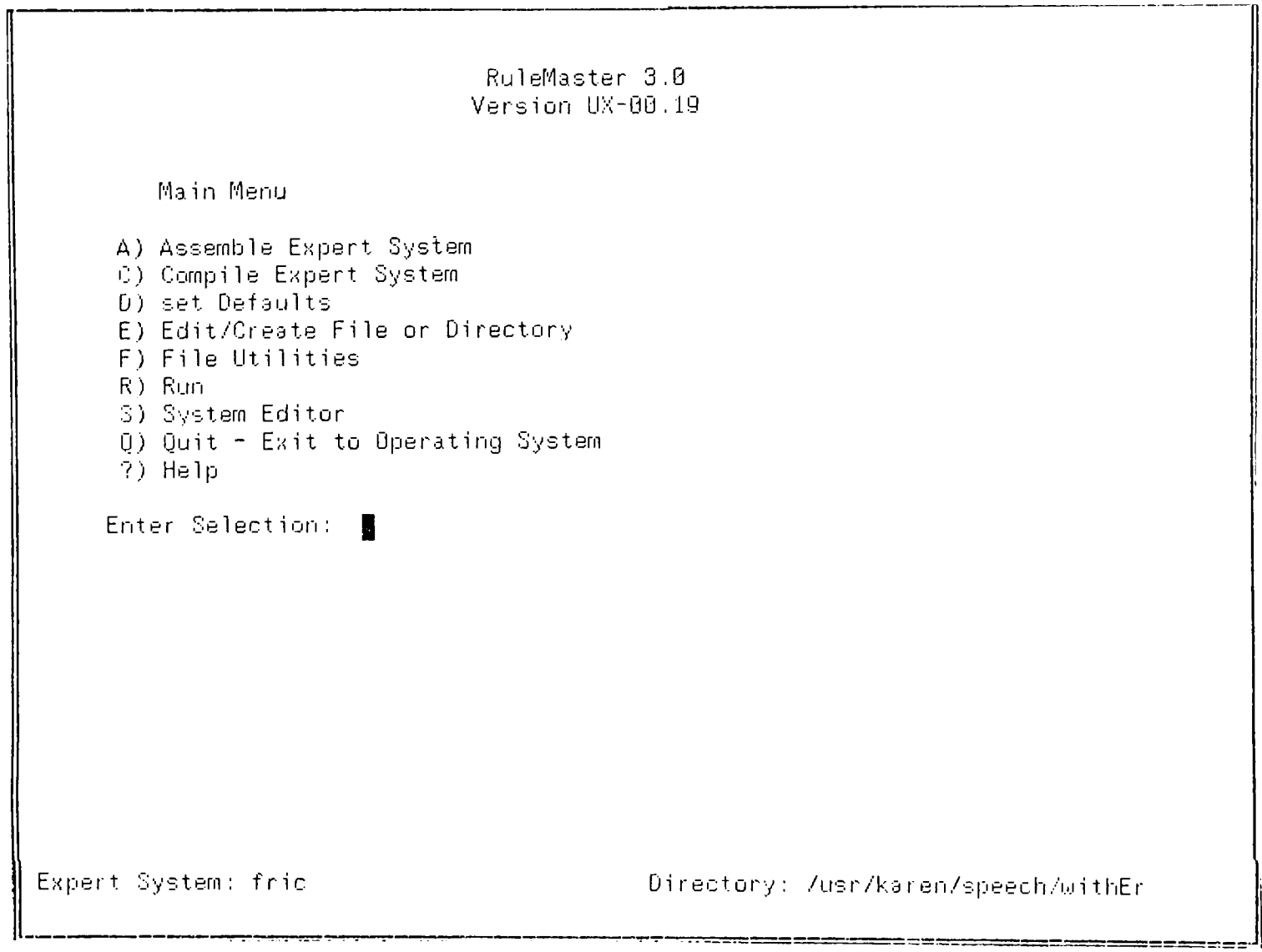


Figure B

RuleMaster Sysed Menu

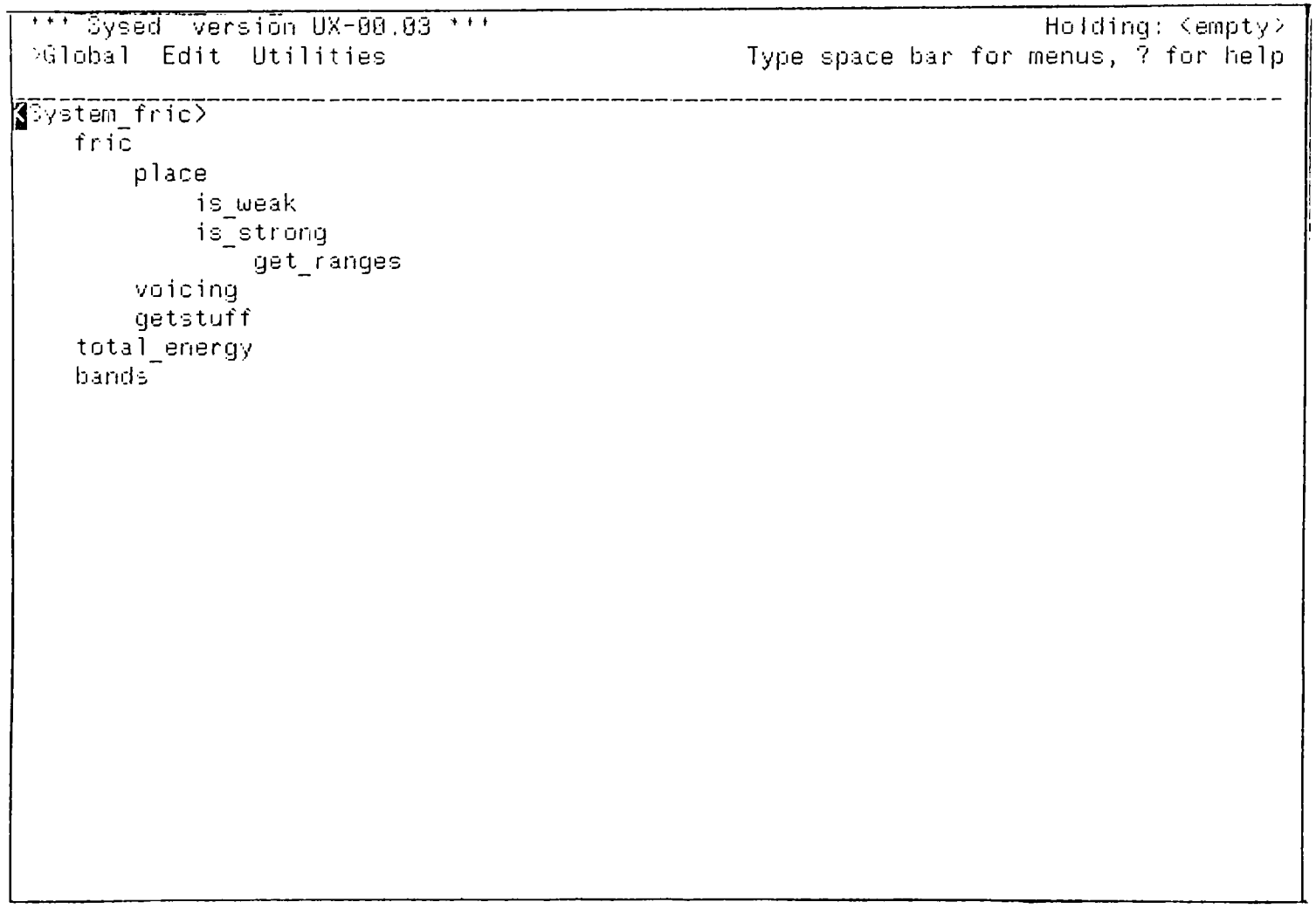


Figure C

large system while the finished product is intended for use on micro-computers.

The RuleMaster system can hasten the development of a rule-based expert system, not only by supplying the control structure, knowledge representation and explanation facilities, but also by including intelligent editors and a rule inducer.

Sysed with Pulldown

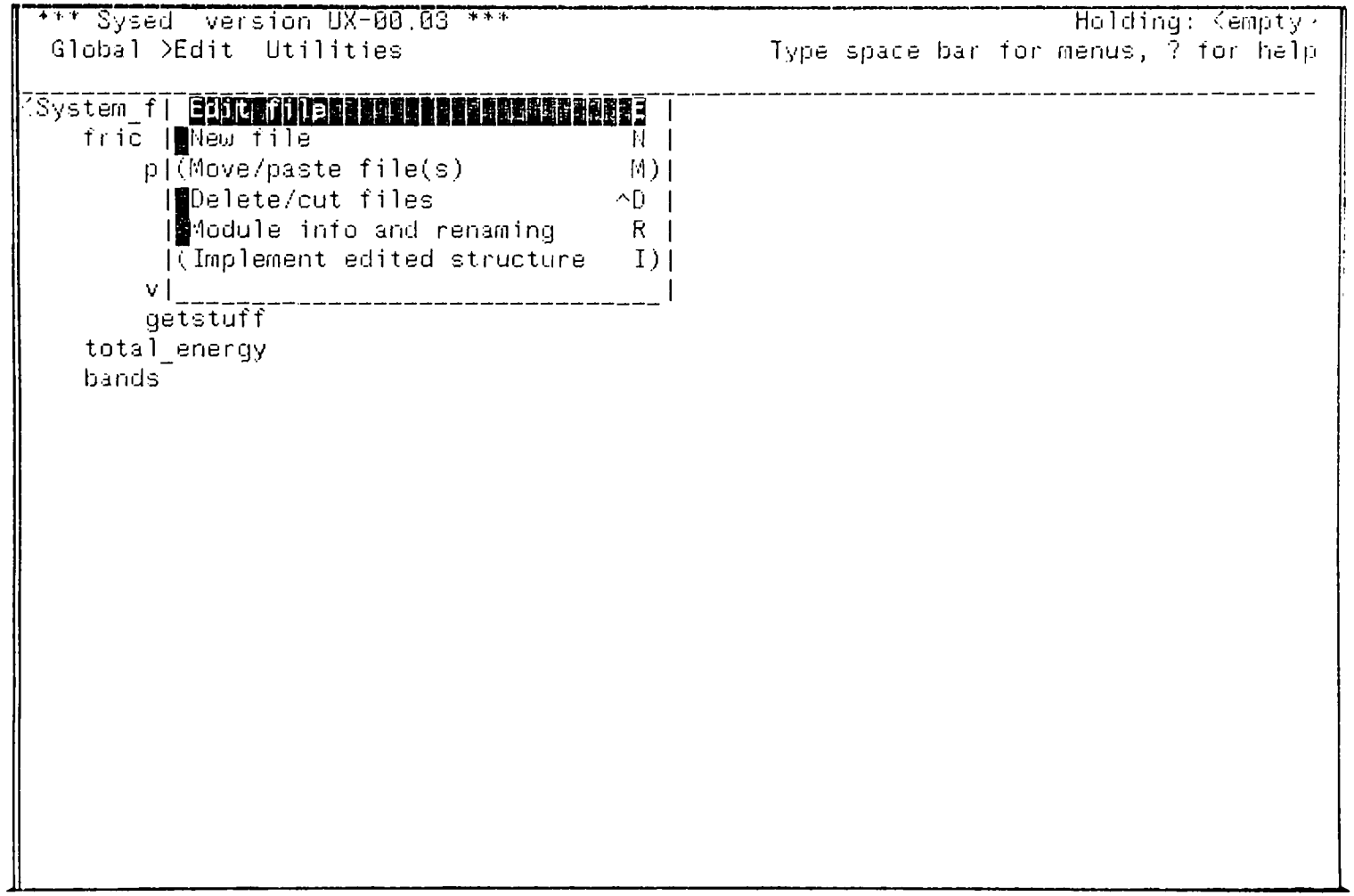


Figure D

Example Table

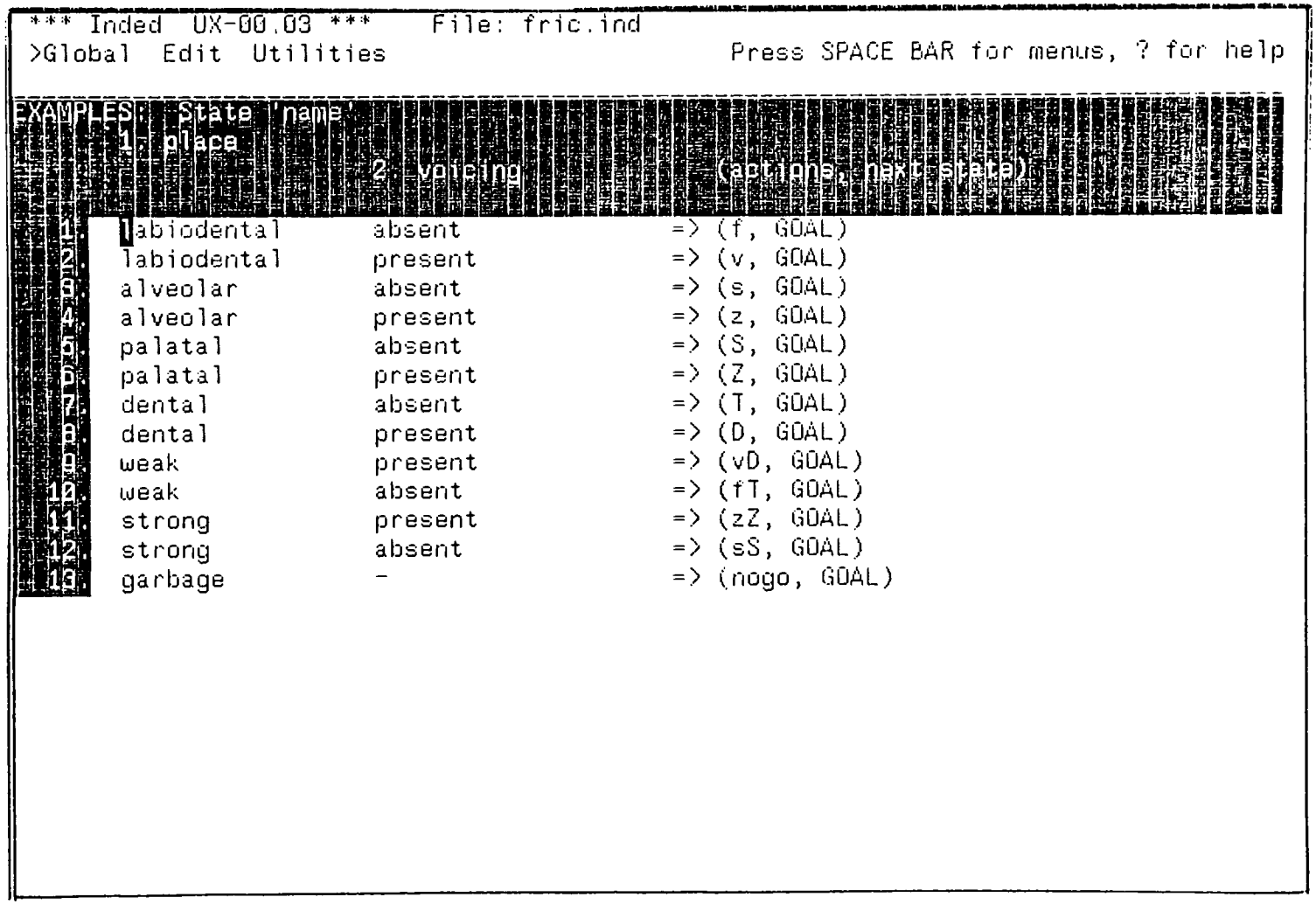
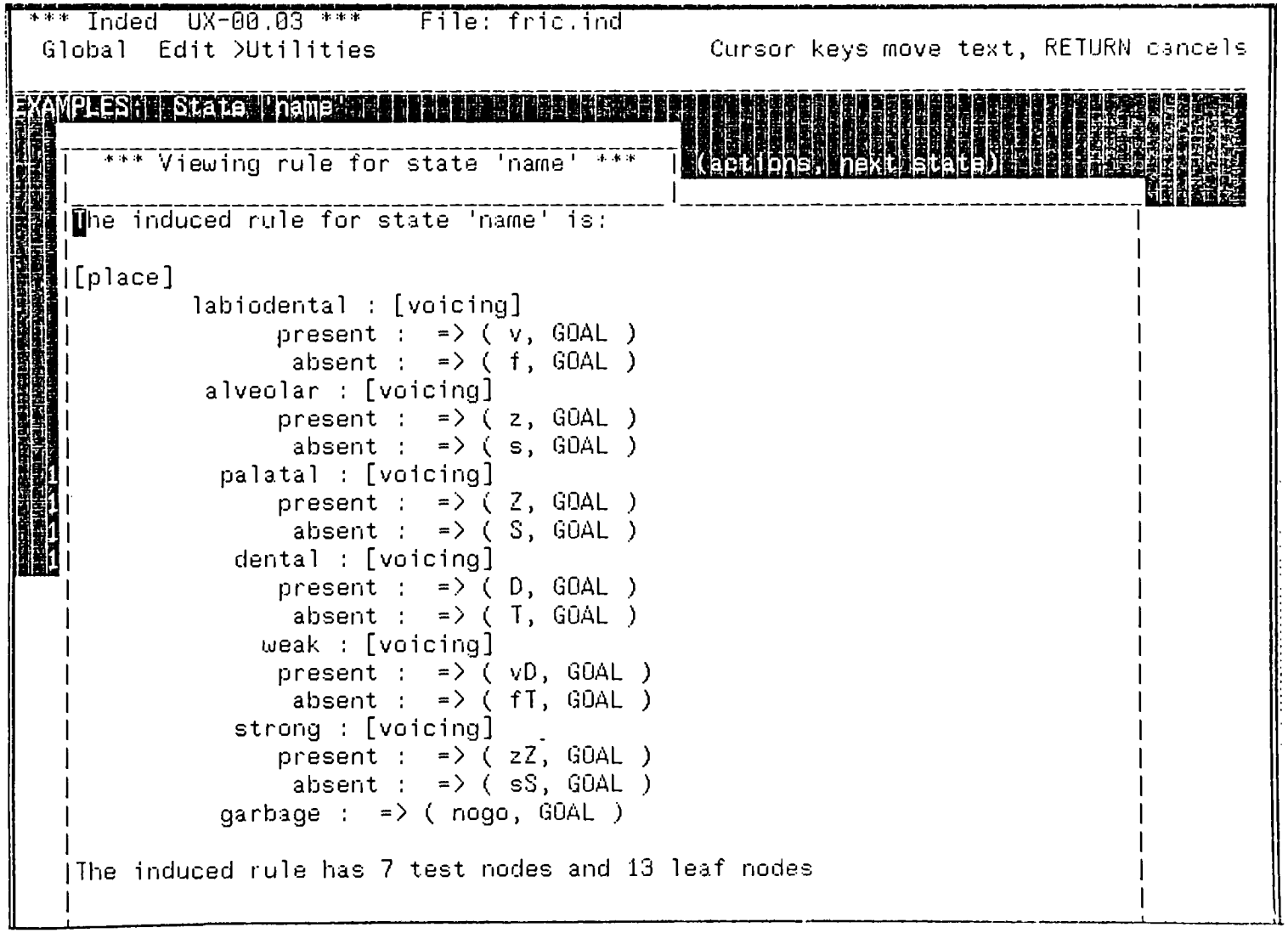


Figure E


*** Inded UX-00.03 *** File: fric.ind
Global Edit >Utilities
Cursor keys move text, RETURN cancels
EXAMPLES: State 'name'
*** Viewing rule for state 'name' ***
(actions, next state)
The induced rule for state 'name' is:
[place]
labiodental : [voicing]
present : => (v, GOAL)
absent : => (f, GOAL)
alveolar : [voicing]
present : => (z, GOAL)
absent : => (s, GOAL)
palatal : [voicing]
present : => (Z, GOAL)
absent : => (S, GOAL)
dental : [voicing]
present : => (D, GOAL)
absent : => (T, GOAL)
weak : [voicing]
present : => (vD, GOAL)
absent : => (fT, GOAL)
strong : [voicing]
present : => (zZ, GOAL)
absent : => (sS, GOAL)
garbage : => (nogo, GOAL)
The induced rule has 7 test nodes and 13 leaf nodes

3. Speech

3.1. Speech Generation

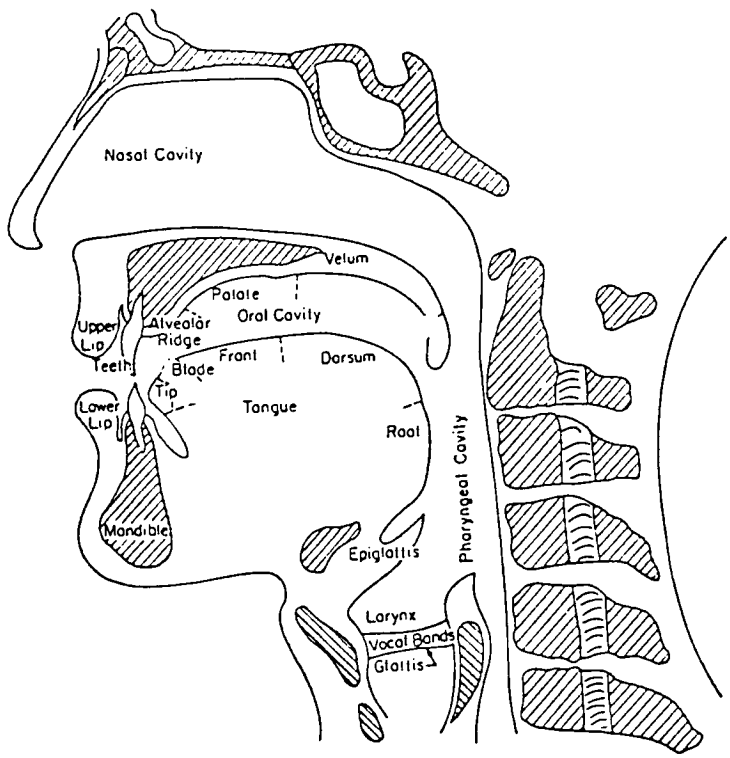
Human speech production can be viewed as the conversion of muscular energy to acoustic energy. The muscular energy is used to induce a pressure change in the vocal tract and set the air into motion. It can then be used to regulate the flow of air or modify the sound waves [10]. After the airflow is generated, it passes through the laryngeal cavity, which consists of the larnyx, vocal bands and glottis, as shown in Figure G [22]. Vibration of the vocal cords by the passing air is known as voicing. The air then passes through the pharyngeal cavity where it can be routed through the oral or nasal cavities. Each of these has its own resonating characteristics.

The nasal cavity is brought into play by lowering the soft palate and forcing air upward. Complete oral closure creates nasals such as "m" or "n" sounds. If the oral closure is not complete, the sound is considered nasalized.

In the oral cavity are the articulators, structures which modulate the air flow [15]. The upper articulators consist of the upper lip, the upper teeth and the entire roof of the mouth including the alveolar ridge, the palate, and the velum. The lower articulators include the lower lip, lower teeth and tongue which is divided into areas called the tip, blade, front, dorsum and root [10]. By employing these structures, the airflow can be altered to create a variety of sounds.

Sounds have features that bear a direct relationship to the articulatory gesture which produced the sound. For example, airflow circumventing obstructions or passing through constrictions causes frication, which can be heard in the noisy quality of an /s/ sound and evidenced in the waveform as a high zero crossing rate. Voicing, positioning the vocal cords to vibrate with the airflow as in a /z/ sound, causes the wave form to be periodic [51].

Because speech is a continuous series of movements of a great number of muscles and other structures, the possibilities for different movements or combinations of movements are practically unlimited. Even when repetitively producing sounds that appear identical, closer



Cross section of head showing principle speech organs

Figure G

examination would show that they are not. As humans do not register minute differences in sounds, sounds with small variations can be classified as the same sound. [29].

3.2. Phonetics

Phonetics, the study of speech sounds, is concerned with the determination of the basic sounds in a language, the symbolic nature of these sounds, and the study of how speech sounds are produced and perceived [15]. Although the number of possible sounds is enormous, the actual number of basic sounds in a given language is quite restricted. In English there are about forty basic sounds, referred to as phonemes. Each phoneme has distinct properties according to the place of articulation, manner of articulation and voicing. These features serve to classify the phonemes.

The place features of a sound refer to the position of articulatory mechanisms during the production of sounds. Labio-dental, alveolar, palatal, velar, dental, and palato-alveolar are terms used to describe the place features of the consonants, implying involvement of the lips and teeth, alveolar ridge, palate, velum, teeth, palate and alveolar ridge, respectively [15].

Manner features are indicators of how the sound is made. A *stop* sound, as in the beginning of the word "to", is produced when the airflow is actually stopped. Pressure is built up and then released. A *fricative* is characterized by turbulence caused by a constriction or an obstruction in the airflow. The /f/ sound in "foo" is a fricative. *Nasals* are created when the nasal cavity is brought into play and can be heard in the consonant sound in the word "no". *Glides* are sounds produced by cavity modulation as found in the beginning sound in "woe".

Many of the basic sounds have voiced-voiceless *cognates*, sounds that have the same place and manner features, but the vocal bands are vibrated when creating one of the cognates. Adding voicing to a voiceless sound creates a new sound. As an example, the /s/ sound is a voiceless fricative with /z/ as its voiced cognate.

The vowel sounds are classified as front, central or back, depending on the position of the highest part of the tongue in the oral cavity during when the vowel sound is produced. Each of these classifications are described as high, mid or low according to the height of the tongue. A high front vowel sound is in the word "bit". A low back sound can be heard in "not" and the /a/ sound in "palm" is a low mid vowel.

Although phonemes have been studied for some time, the greatest advances have been made since the 1940's when the sound spectrograph was invented. This device makes visible records of the fundamental dimensions of speech: frequency, intensity and time. This record, called a spectrogram, is produced using frequency as the vertical axis and time as the horizontal axis. Variations in intensity are depicted by the darkness of the pattern.

A spectrogram displays indications of the types of modulation in speech (figures H and I). A stop is viewed on the graph as a pause or silent spot in the speech flow and is frequently followed by a burst of energy, whereas a wide range of randomly placed frequencies indicates a fricative. Vocal band modulation, leading to voiced phonemes, is sometimes evidenced as a bar of intensity across the low frequencies. Vowels, which are a combination of vocal band and cavity modulation are indicated on a spectrogram by horizontal resonance bars called formants. Relationships between these formants are clues to the identity of the individual vowels.

Trying to determine the utterance recorded in a spectrogram is not a simple task. Speaking produces a complex acoustic signal that contains extra-lingual material as well as the linguistic message. Speaker attributes such as physiology, sex, age, emotional state, and even whether the speaker is suffering from a cold, may be reflected in the speech signal. To complicate matters further, speech signals will differ not only from speaker to speaker but also in repetitions of the same utterance spoken by the same speaker [26].

In addition, in continuous speech basic sounds are combined, causing a blurring of boundaries and properties of the individual sounds. This effect is referred to as coarticulation. In analyzing connected speech, one must differentiate coarticulatory effects from speaker variability [26]. Each phoneme has unique articulatory and acoustic properties which change with the phonetic environment. This overlap of phonetic information in the acoustic signal makes the spectrogram difficult to interpret and prevents employing a simple template matching solution [48].

spectrogram of utterance

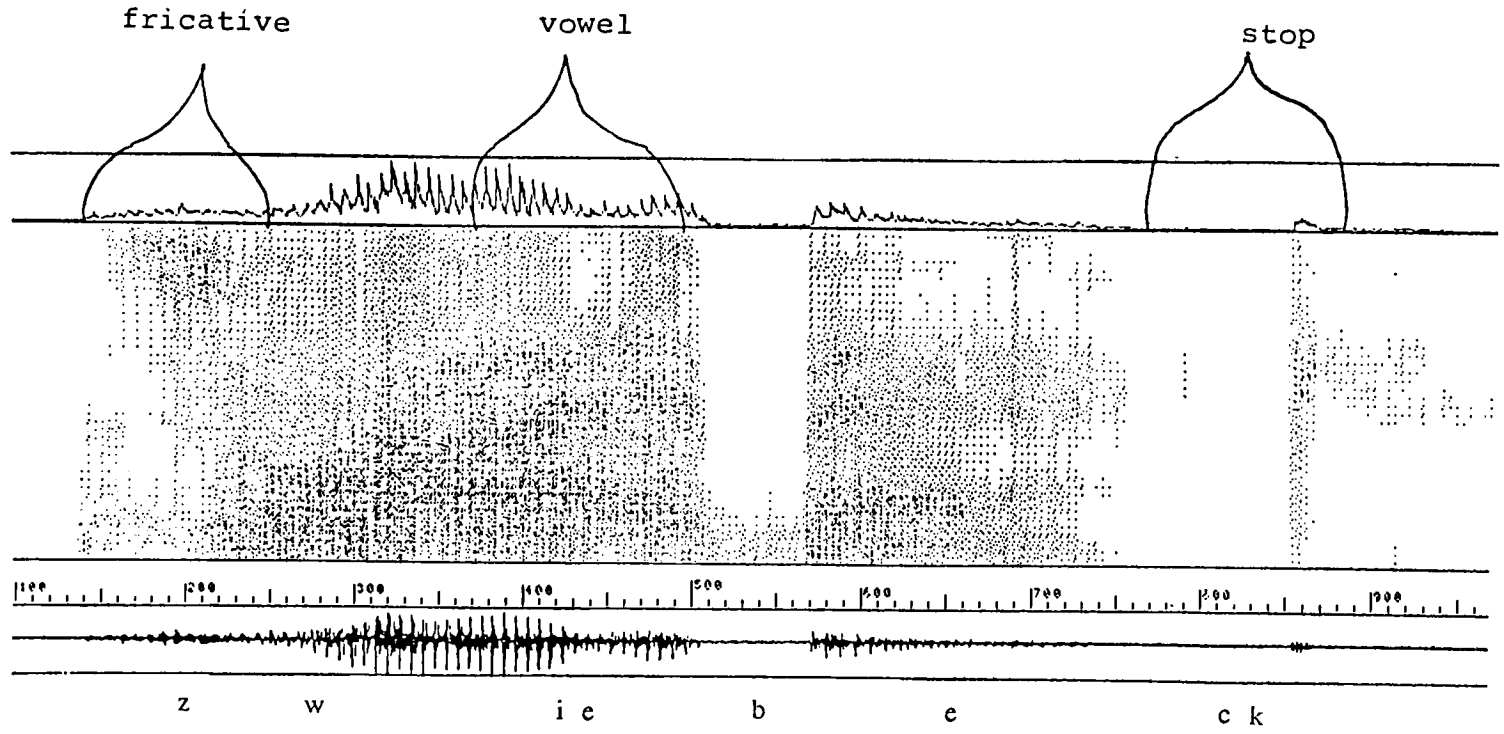


Figure II

spectrogram of utterance

six examples

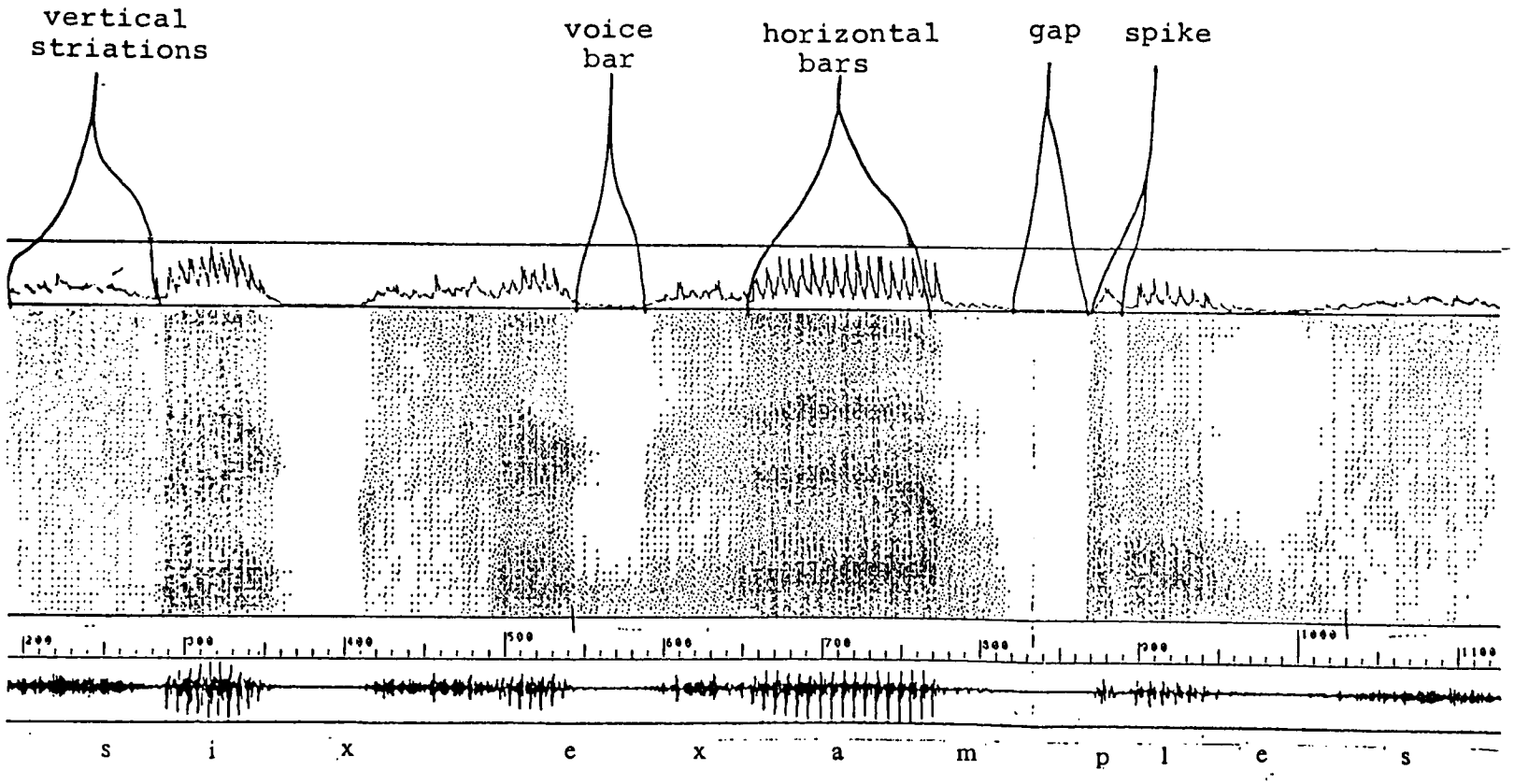


Figure I

3.3. Reading Spectrograms

A necessary step in reading a spectrogram is to translate the visual representation of sound into a string of phonemes. This task is extremely complex as there are a myriad of acoustic cues whose significance must be interpreted with regard to other evidence [48]. Some cues indicate coarticulatory effects, but others indicate extra-lingual information.

Rhythmic features of speech, such as intonation and stress, affect the semantics of the utterance. These linguistic concepts have acoustic manifestations found in fundamental frequency, overall amplitude, spectral balance and sound duration [26]. Correlations between linguistic categories and acoustic events are difficult to make because they are situationally dependent. As an example, stress is the amount of muscular energy in the articulatory movement. Acoustic measurements of speech energy are useful indicators of stress, but only when the energy of a given sound is compared to the energy of the same sound under different circumstances, as in a different utterance by the same speaker [27]. Understanding linguistic concepts is not necessary to transcribe a spectrogram into a sequence of phonemes although it is useful when discerning the semantics of a phoneme sequence [40]. Therefore this information should not be used to identify phonemes, but should be employed when attempting to identify the words present in a sequence of phonemes.

To determine the phonemes in a spectrogram, boundaries that pinpoint changes in spectral composition must be determined. Next these segments should be labeled by classifications such as fricatives, stops and vowel-like sounds [17]. A running estimation of the place of articulation is sometimes carried out for the entire utterance before trying to distinguish actual phonemes. The information found in consonant segments tends to be more reliable than that found in vowel segments as vowels suffer more from coarticulatory effects [47]. Therefore it is wise to identify the consonants first.

In the process of determining the phonetic class of a sound and narrowing that description until a specific phoneme can be hypothesized, a good deal of domain specific knowledge comes into play. Each phoneme has a characteristic pattern, regardless of speaker, that can

be altered by coarticulatory effects. Therefore, it is necessary to analyze the presence of certain acoustic cues to detect the presence of a particular phoneme.

The visual phenomena of a spectrogram include spikes, gaps, horizontal bars and vertical striations (figure I). These phenomena relate directly to the physical manifestation of muscular energy during speech production. For example, when the airflow is stopped, a gap will appear in the spectrogram. Eventually the air is released causing a spike in the graph. Vertical striations indicate frictional modulation, whereas vocal modulation can be detected by the presence of a voice bar or from periodicity in the waveform [37].

Stop sounds can be voiced or unvoiced and are characterized by a pattern where a gap is followed by a spike. Fricatives also have voiced-voiceless cognates and are identified by irregular vertical striations which are called a fill. These fills are generally wider than the spike fill for stop sounds.

Vowels and vowel-like sounds are characterized by horizontal resonance bars indicating vocal cord and cavity modulation. Although each vowel sound in isolation has a characteristic formant pattern, coarticulation changes these basic patterns.

Many rules about phoneme spectral patterns are known, but to be used effectively the reader must know when to apply these rules and when to ignore them. Visual features may occur in spectrograms for a variety of reasons. For instance, the duration of a sound can be a clue to the identity of a phoneme, but can also indicate stress, voicing or other information. Some vowels are short by nature and others are short because of their environment. For example, a vowel is shorter when it precedes a voiceless sound than when it precedes a voiced sound [37]. Stressed vowels tend to be longer in duration than unstressed vowels, and if a final syllable is unstressed, its duration is quite short [30]. Also, certain consonant clusters significantly abbreviate or totally delete a member consonant. While a /t/ sound is classified as a stop, in the word butter the /t/ sound is pronounced as a fast stop and closure is not complete. Its characteristics are significantly different from the stop /t/ and it is known as a flap.

With all the problems inherent in reading a spectrogram, the multitude of rules which must be selectively applied, there are those who have hypothesised that it is not possible to train a person to read them effectively [33]. To settle this controversy, an experiment designed to assess a person's ability to read unknown utterances from spectrograms was conducted at Carnegie-Mellon University in late 1977 and continued in early 1978 [13]. Victor Zue, a leading researcher in the field of speech recognition who began a systematic study of spectrograms in 1971, was chosen as the subject who would attempt to identify utterances from their spectrograms.

Zue's task was to identify the phoneme strings represented by 23 spectrograms of utterances by two male speakers. First he identified segment boundaries and then labeled them phonemeically. Three trained phoneticians also transcribed the utterances, but they did so by listening to them. A segment was said to exist when two of the three phoneticians agreed on its existence. When Zue was not sure of a phoneme label, he gave a second choice. His labels agreed with at least one of the phoneticians 85% of the time. As the average agreement among the phoneticians was only about 90%, Zue's performance was a strong showing of his ability.

An interesting point was made when the same experiment was run using nonsense words rather than English sentences. Zue's performance on nonsense words was 90% correct, indicating that high-order knowledge about the syntactic structure and semantics of English is not needed to read a spectrogram [49].

Observation of Zue's performance led to the conclusion that phonetic segments can be identified by characteristic visual patterns. His extensive knowledge of the effects of coarticulation on these patterns was instrumental in the interpretation of the spectrograms [13].

As further evidence, five spectrogram readers were trained with a 13 week course in acoustic phonetics. The first ten weeks concentrated on learning static characteristics of speech sounds, while the final three weeks stressed the rules governing concatenation of sounds in continuous speech. When tested on spectrograms of five sentences, the group iden-

tified phonemes with an accuracy of 80% [49]. As Zue and Cole stated in their presentation to the 1979 International Conference on Acoustic Signal and Speech Processing:

> Our experiments demonstrated that phonemes are accompanied by acoustic features that are recognizable on a speech spectrogram, and that with sufficient training it is possible to learn enough about these features, and the modifications they undergo in fluent speech, to read a spectrogram of an unknown utterance [49 p.118].

This evidence would lead one to believe that reading spectrograms is possible, although difficult. The more knowledge one has about characteristic patterns of phonemes and the changes that are caused in these patterns by coarticulation, the more feasible the task becomes.

Although phonemic recognition is a subjective task, the utterance can be identified even from the imperfect transcription. In the experiment using Victor Zue as a subject, the phoneme sequences were examined by a linguist who was able to identify all but 5 words from the fifteen utterances. Therefore using phoneme identification as a step in a speech recognition project provides a reasonable platform from which a linguist can do word recognition.

3.4. Automatic Speech Recognition

Automatic speech recognition (ASR) was attempted as early as 1948 when ASR devices were doing crude voltage analysis of the speech signal. Even with this primitive approach some words and short phrases could be recognized. Many theories were explored in the Advanced Research Project Agency (ARPA) speech understanding projects in the early 70's. Common methodologies found in these projects include template matching, stochastic modeling, and probabilistic parsing [35].

The template matching approach chose a basic unit, frequently a word, and then matched features of the sound spectrum against values stored in memory. Since speech signals are highly speaker dependent, some systems used a training phrase from which parameters were extracted so templates could be altered for each speaker. Because of the memory requirements for the stored templates and the consideration of search space size, systems

using this approach could only recognize severely restricted vocabulary.

Stochastic modeling is a flexible general method for making decisions based on incomplete or uncertain information. Using a probabilistic model, conjectures are made of the desired knowledge (a phoneme string) and various techniques are used to match these conjectures against the observed data (acoustic phenomena). One application of this method finds the word string which is most likely to have produced the acoustic phenomena [3]. Again, a restricted vocabulary is needed to reduce the search space.

In addition to restricting the vocabulary, the grammar acceptable to the recognition system can be constrained. When the grammar of the language is known, a predictive parsing technique can be used to drive the recognition system. This technique uses previously recognized phrases and domain knowledge to build expectations about what the speaker is likely to have said [20].

Some of these uses of knowledge are predictive (top down) and others are bottom-up. The former uses a priori knowledge as an expectations driver whereas the latter is a data-driven approach. Both of these methods have their weaknesses. In a pure bottom-up system, erroneous data can introduce irrevocable sources of error, while in a pure top-down system there may be no clear way to decide among alternatives. These strategies need to interact to generate and discard possibilities. Therefore, an automatic speech recognition system needs to integrate these approaches while maintaining a wealth of knowledge about speech.

The usual approach to speech recognition has been top down. Only limited efforts have been devoted to bottom up analysis of the speech signal, although current knowledge indicates that the signal contains a great deal of information [20]. Information obtainable from speech signals now includes numerous methods for formant extraction, autocorrelation techniques, zero-crossing density counts and linear predictive analysis. The upper levels of a speech recognition system need to use the information gleaned from a low level analysis of the speech signal to guide the hypothesis generation.

A choice of methods is available for analyzing a segment of the speech signal. Pattern matching, a computationally driven approach, preserves spectral details and statistically determines distance measures which numerically categorize the differences between the spectral segment and templates. With feature extraction, acoustic characteristics are interpreted according to features that have perceptual or linguistic significance. Examples of feature extraction are relating the amount of low energy to nasalization or the amount of total energy to stress [15].

Various architectures have been used in the design of speech recognition systems. One ARPA project, Hearsay II, made great strides for the AI community with its unique control structure. The system was comprised of a set of parallel asynchronous knowledge sources which communicated via a global structure called a blackboard. A knowledge source (KS) could propose a hypothesis to a higher level KS via the blackboard or it could predict or verify the hypothesis of a lower level KS [28]. The blackboard provided a good model of multilevel or opportunistic reasoning.

Levels of knowledge included in a speech recognition system include a signal segmenter, a segment labeler, and a word hypothesizer that interacts with a scheme to limit the number of probable words. Knowledge about syntax and semantics is needed as well as knowledge about intonation and rhythm. The latter, referred to as prosodics, is included to predict or confirm proposed syntactic structures. Pragmatic information can be used to determine if a possible sentence is appropriate in context [42].

Speech recognition systems all constrain the problem space in some manner. Besides the vocabulary and grammar restrictions previously mentioned, some systems do not deal with continuous speech; the speaker is required to enunciate clearly and pause between words. This action segments the incoming input signal into discrete units and a very difficult part of the continuous speech recognition problem, word delimitation, is circumvented.

Limitations on the number of speakers a system can handle have also eased the problem of recognition. If only a single voice will be used, the amount of variance that can be

expected in the speech signal is reduced. Even with one speaker repeating an utterance, there can be a great deal of variation in the signal, but when multiple speakers are introduced, the differences in vocal tract length and shape, as well as variations in voice pitch and speaking rate will act to complicate the problem by creating monumental variances in the speech signal.

Creating a speech recognition system is a complicated task that is usually simplified by restricting the vocabulary, grammar or number of speakers. The increase in problems encountered by a system removing any of these restrictions is overwhelming.

4. Fric

4.1. Recognition System Description

The project with which Fric is associated is a speaker independent, large vocabulary, continuous speech understanding system. The architecture proposed for this system is illustrated in figure J. The system attempts to combine top-down and bottom-up methods to make the best possible use of available knowledge.

The bottom-up control starts by processing the input signal through Speech Tool, an interactive speech research tool created at Speech Research Systems in Rochester, New York. The signal processing software provides waveform information in graphic representation (Figure K) which is useful in segmenting the signal. As segments are identified by manner features, each segment can be passed to recognizers specializing in a particular classification. A graph of the zero-crossing counts can be used to determine which segments would be classified fricative as a high zero crossing count is indicative of the noisy sounds associated with fricatives. The beginning and end times associated with each segment are determined by viewing the ticks graph which shows time in milliseconds. The collection of recognizers is shown in Figure J as the intermediate level phoneme builder. Fric is one of these recognizers and will identify segments that are fricative in manner. (Figure L)

After the phoneme string is identified, a word builder (Figure J) attempts to find word boundaries and identify the words in the utterance. This will be achieved by a dictionary lookup system currently under development.

Feature extraction is a method that allows the selective focusing on specific qualities of the speech signal. Spectrogram readers use this approach when attempting to transcribe an utterance represented graphically. Combining feature extraction with the knowledge used to read spectrograms provides a feasible methodology for phoneme identification that can reduce data without losing necessary information. This approach to the identification problem was used in Fric.

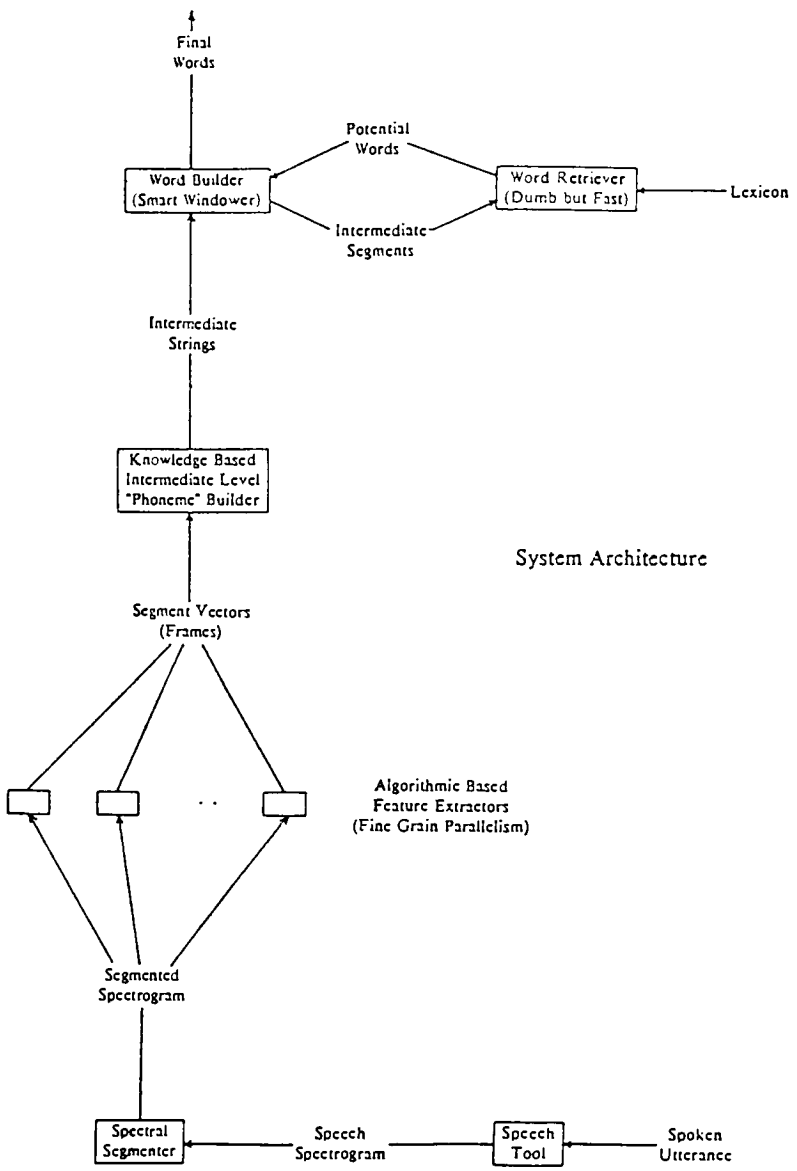

Final
Words
Potential
Words
Word Builder
(Smart Windower)
Word Retriever
(Dumb but Fast)
Lexicon
Intermediate
Segments
Intermediate
Strings
Knowledge Based
Intermediate Level
"Phoneme" Builder
System Architecture
Segment Vectors
(Frames)
Algorithmic Based
Feature Extractors
(Fine Grain Parallelism)
Segmented
Spectrogram
Spectral
Segmenter
Speech
Spectrogram
Speech
Tool
Spoken
Utterance

Figure J

1 basic sound wave

2 transformed sound wave

3 ticks (time in milliseconds)

4 sum file

5 zero-crossing count

Speech Tool Display

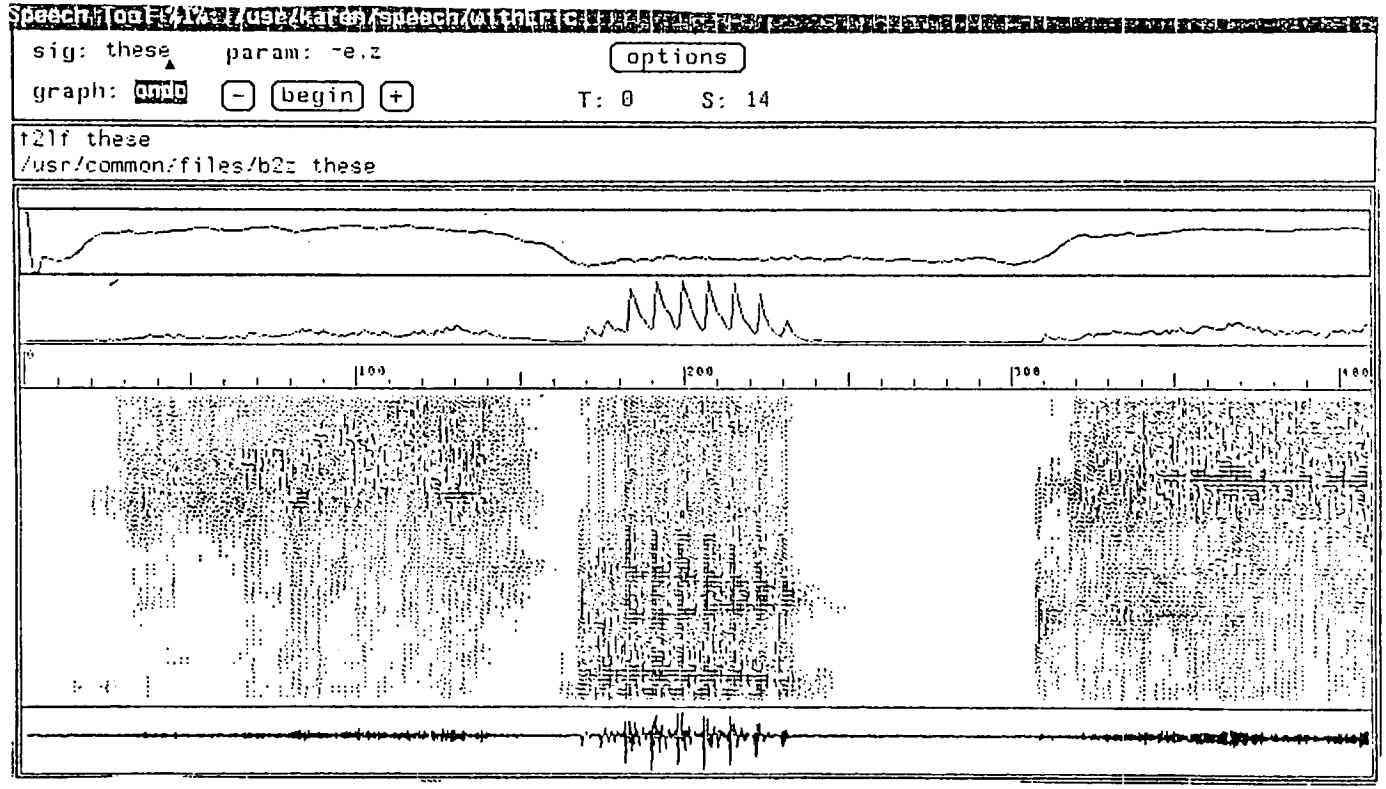


Figure K

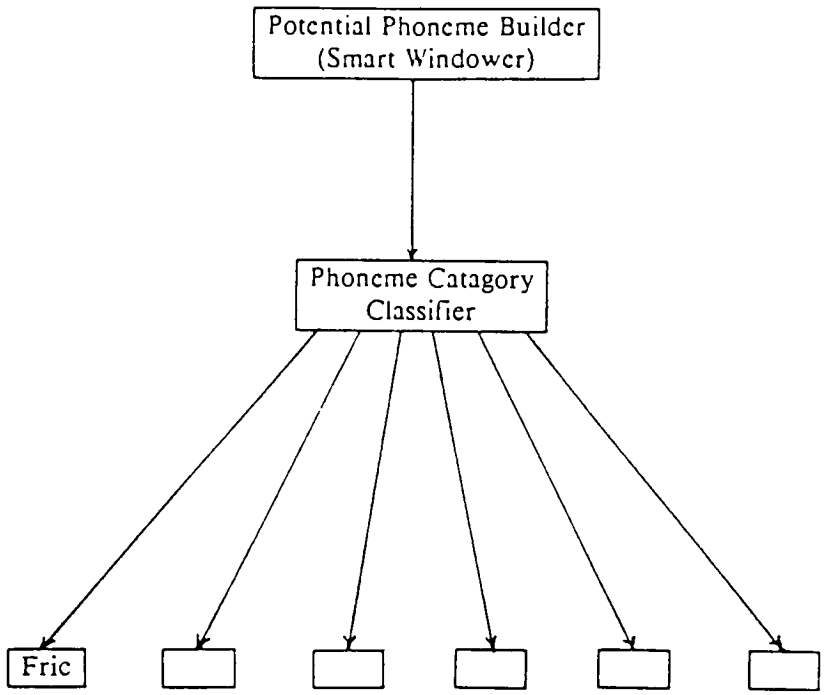


Figure L

4.2. Knowledge Engineering

Trying to find an expert spectrogram reader is a difficult task. A very few books, manuals and papers exist on the methodology [37,17,47], and there is no exhaustive description of the process. There is an incredible amount of written rules concerning coarticulation and its effects on a visual representation of speech.

Although no true expert spectrogram reader could be found for this project, two experts in speech were available. Robert Houde is a recognized authority in speech science

and speech signal processing. He has been working in the domain of speech since 1957 and received his Ph.D. in communications science from the University of Michigan in 1967. Dr. Houde founded the Center for Communications Research in 1970, an organization researching concerns of the deaf. In 1983 he started Speech Recognition Systems, Inc., and has been attempting to build a speaker independent continuous speech recognition system. Although Dr. Houde has been working with spectrograms for many years, he has not made a practice of reading them.

The services of Dr. James Hillenbrand were also available. Currently a research scientist for the RIT Research Corporation, Dr. Hillenbrand received his Ph.D. in Speech and Hearing Science in 1980 from the University of Washington. He has worked in this field since 1970 and has acquired a great deal of knowledge about acoustics, the branch of physics dealing with sound. This was evidenced by the speed with which he was able to learn to read spectrograms. Although this practise was not part of his work before employment with RIT Research, Dr. Hillenbrand began studying spectrograms for this project and was soon able to discern manner features by the spectral manifestations of the physical actions which created the sounds.

Having no previous knowledge of linguistics, phonetics, spectrograms and very little about automatic speech recognition myself, it was first necessary to read extensively about these subject areas. There is no lack of information about any of these topics; consequently choosing appropriate reading materials was crucial. Dr. Houde was helpful in suggesting books, articles and authors. Once the background research was well under way and the basic concepts of the domain were understood, it became time to start the interviewing process. The first meeting, held with both experts, was conducted as an informal exchange of questions and ideas. Knowing that the task of recognizing all the phonemes was well beyond the scope of a Master's thesis, it was decided, upon Dr. Houde's recommendation, to attempt recognition of the fricatives (Figure M).

Fricative Phonemes				
Symbol	Sound	Voicing	Place	Strength
f	fat	voiceless	labio-dental	weak
v	vat	voiced	labio-dental	weak
T	thin	voiceless	dental	weak
D	then	voiced	dental	weak
s	yes	voiceless	alveolar	strong
z	zoo	voiced	alveolar	strong
S	show	voiceless	palatal	strong
Z	azure	voiced	palatal	strong

Figure M

Because of coarticulation, recognizing phoneme strings can create a problem where each phoneme must be identified in order to recognize its neighbors. Consonants suffer the least from coarticulation and so are a good place to start the recognition task. Two of the manner classes of consonants are fricatives and stops. Since Victor Zue is designing a knowledge based system to recognize stop sounds [50], this system attempts recognition of the fricatives.

Brief meetings with the experts helped clarify readings and questions that arose while researching the phoneme identification problem and prototyping the system. The next formal meeting held was a demonstration of the first prototype of Fric. This preliminary version of Fric assumed that all fricatives would be positively identified and that the system would be interacting with an expert looking at a spectrogram. This totally interactive system guided the expert in doing feature extraction on the spectrogram. The design aim of Fric was to replace questions to the expert with automatic routines that could answer these questions by analyzing the information provided by Speech Tool. It was not expected at this time that fricatives would be identified accurately, but it was hoped that Fric would function well enough to demonstrate the approach was valid. Both experts and a number of interested parties spent about two hours interacting with the preliminary version of Fric. In that time the word "shackles" had the beginning and ending fricative sounds correctly identified and the word

"frazzled" had its beginning sound correctly identified, although the z sound was incorrectly labeled as a voiced /th/.

One problem that was identified at this point was that of unambiguous terminology. For instance, when asking about the amount of total energy in the sound segment, the system asked, "Is the sound weak or strong?". Weak and strong have phonetic connotations (Figure M) that caused the expert to think the system was asking about something other than how much energy was in the signal. It is possible to have a weak fricative display a large amount of energy and it is possible to have a strong fricative with a small amount of energy although this is not usual. Revisions of Fric were made with attention to the meanings of terms used. Had the system been intended to be totally interactive, it would have been useful to include definitions of terms so that the system user would be sure to understand the questions and the rules.

Because much of the time in this session was spent explaining parts of the system to observers who were not involved with the project, it was quickly decided to limit the number of people in an interview. Subsequent interviews were held with only one of the experts at a time. During these sessions the expert read spectrograms and was questioned about his reasoning techniques in an attempt to formulate the rules and control strategies needed for Fric. These sessions were taped to minimize loss of any information.

The next interview took place at Dr. Houde's place of business where he was recorded reading spectrograms for about an hour and a half. This time the interview was extremely productive. Knowledge about reading spectrograms was gained as well as guidelines about choosing sample utterances. It was necessary to separate similar utterances to prevent Dr. Houde from template matching. His work tends to use template matching as a fundamental concept and though Fric's strategy was to avoid that approach, Dr. Houde tended to use it whenever possible.

When reading spectrograms Dr. Houde would frequently use knowledge that would not be available to Fric. He knew the focus was on fricatives, and this slanted his answers. He

also used his knowledge of the English language to fill in gaps in the transcription of the spectrogram. The following are quotes of the interview with Dr. Houde on 4/7/86.

> And there's a fricative here because there are fricatives everywhere.

> So maybe this says "if this", and this is all front vowel too, so this is strong so this could be "is". Maybe "if this is iii" and I could put a t on the end just because it would make linguistic sense. "if this is it", but I don't have any other reason to put the t on it. I don't think that is a t.

When the sample utterances were single words Dr. Houde's ability to phonetically transcribe the utterance diminished. This can be attributed to the inability to use any higher level knowledge. It was decided that the next set of examples should be words strung together without making sense, so that the higher level knowledge sources could not be used.

Dr. Houde's requests for subsequent sessions included a scale of 6 KHz on the spectrogram rather than the 4 KHz in use because many strong fricatives (/s/ or /z/) contain most of their energy above 4 KHz. Dr. Houde could discern the presence of these strong fricatives by the amount of total energy, which is shown in another graph, the sum function, even though there was no corresponding darkness on the spectrogram. Also requested was a print of the spectrogram made with the time scale set so individual pitch periods could be seen. When the compressed time scale made it impossible to detect periodicity, Dr. Houde was unable to tell if a sound was voiced.

The valuable lesson learned from this session was that although voicing may have many effects on a sound, one criterion, periodicity, was enough to detect voicing. During the background research many of the rules concerning voicing used its presence as a prerequisite condition leading to a particular consequence. What the system needed and now had was a rule that said *if a certain acoustic phenomena (periodicity) is present,* then one can conclude voicing is also present. Consequently later efforts were concentrated on finding ways to discern the

place of articulation.

Subsequent interviews required reading spectrograms that were designed to elicit specific information. The evolving set of rules was:

(1) Discriminate between weak and strong fricatives.

(2) If strong, look at the pattern to see where the energy is concentrated.

(3) If weak, take a guess.

(4) Look at voicing information.

Dr. Houde could not discriminate among weak fricatives by any method other than template matching. If a weak fricative had been previously identified, he would compare its spectral composition with the segment under question. As template matching is not a desirable method in a speaker independent system and expert spectrogram readers have no need of templates, the next interview, which was held with Dr. Hillenbrand, attempted to find rules that would help to discriminate between weak fricatives.

Although these rules were eventually learned from Dr. Hillenbrand, lessons about interviewing were learned first. Interviews with Dr. Houde usually took place in a rather secluded room with his secretary fielding phone calls. The number of interruptions was minimal and the time spent was quite productive. On the other hand, Dr. Hillenbrand was interviewed in his office, where interruptions were so frequent that it was difficult to maintain a stream of thought. The lesson learned is that knowledge acquisition is a difficult task requiring concentration and quality time of both the expert and the knowledge engineer.

4.3. Fric - System Development

Due to the nature of spectrogram reading techniques and the availability of RuleMaster, the decision for a rule-based knowledge representation was made. The choice of a control method was flexible as RuleMaster allows combinations of various control mechanisms. The top level modules of Fric perform forward chaining to identify the fricative. These modules call subroutines that backward chain to determine the place of articulation and whether or

not voicing is present in the sound, as these two facts are sufficient to identify a fricative phoneme.

The preliminary version of Fric (appendix 1) had a top module which called two child modules, place and voicing. The values these routines returned were used to distinguish among the possible choices. It was assumed in this version that both the place of articulation and voicing could be discerned with certainty, so there was no attempt to accommodate uncertain situations.

In this version, it was also assumed that there would be no conflicting cues as to the place of articulation. This is a very naive viewpoint, and appendix 2 shows the modification of the place module with conflict resolution. The conflict dealt with concerns about the amount of total energy in the sound and how it relates to the classification of the fricative as strong or weak. The basic expectation was since /s/, /sh/ and their voiced cognates are considered strong fricatives, the speech signal will have a large amount of energy. The weak fricatives, /f/, /th/ and their voiced cognates, should have small amounts of energy. Unfortunately, if an /s/ sound is said softly, the amount of energy inherent in the sound can lead one to believe it is weak. Sounds at the end of an utterance are often said with less energy and these easily confused this first system.

The decision as to whether or not a sound was voiced was based on the existence of a voice bar or periodicity in the waveform, although conditions based on durational information were also included. In later versions when knowledge had been obtained about the relationship between periodicity in the waveform and voicing, the durational conditions in the module voicing were omitted.

This removal of conditions from the example table exemplified an important issue. Although many rules are known about phonemes after the sound has been recognized, not all of these rules can be used to identify the phoneme. Duration is one dimension of a phoneme that can be altered by a multitude of causes. Only after the phonetic environment and the sound have been identified, can one assess why a phoneme is long or short. Durational infor-

mation can be misleading when the context is not fully understood.

Given this phenomena, it is more easily understood why the next version (appendix 3) seems simpler that the first ones in some respects. At this point, rules that had proved to be irrelevant had been removed. This version did start to deal with the fact that not all phonemes can be identified with certainty. When Dr. Hillenbrand and Dr. Houde tried to read spectrograms, they frequently had trouble differentiating between the two possible places of articulation for weak fricatives. Assured by Dr. Houde that an effective algorithm for detecting periodicity in a waveform existed, uncertainty was not allowed for in the voicing module, although it was accommodated in the place module. Consequently, the top module could now identify the phoneme as a weak unvoiced fricative, a weak voiced fricative, or positively as one of the eight fricatives.

Because Fric was designed to be an automatic system when finished, not a great deal of time was spent with the explanation facility. Enough information was included to aid in debugging, but when parts of an explanation were omitted (appendix 4) and there was seemingly no way to remedy the problem, the explanation was left incomplete.

In order to make Fric start answering its own questions, two instances of the Radial interactive ask facility were replaced with calls to outside C routines (appendix 5). Written by Eric Luce, these routines were interfaced with Fric by means of primitive modules. Because of these outside calls, this system now needed information such as the root name of the file containing the utterance (Speech Tool output files for one utterance only differ in their extensions to the basic file name) and the limits on the time segment under question. A new top level module, get_stuff, was added to elicit this information from the user.

These automatic sections of code introduced new possibilities for error and uncertainty. To help compensate for these, the main module's example table was expanded to include alternatives for when discrimination between strong fricatives could not be made and when C routines returned error messages, as when attempting to access a non-existent file.

The place module first determines if the sound is a weak or strong fricative. This was accomplished by a C routine that looked at the energy sum function to determine the amount of total energy in the segment. It was the intention of the control strategy to have *weak* defined with a low enough threshold to prevent any strong sound from being misclassified. If a weak sound was erroneously labeled *strong*, there are later investigations into spectral shape that would detect this error.

If the classification is strong, another routine used the spectrogram file to determine where the concentrations of energy were to be found. This routine accepts two numeric parameters that are measures of 100 Hz *bins*. These parameters are used to determine the boundaries of the acceptable areas of concentration. Differences in vocal tract configurations cause frequency distributions to differ with individual speakers, changing the areas in which these concentrations would be expected. The C routine did a comparison of the energy contained within the area delimited by the two parameters with the energy contained above the higher parameter and returned a value that indicated the area containing the larger amount of energy.

If the fricative is classified as weak, the user of the program must then supply answers to formant tracking questions to identify the weak fricative. This was accomplished by running Fric and Speech Tool in separate windows of the Sun so the user could refer to the graphic displays of the signal in order to extract the necessary data to answer Fric's questions.

Once the call to the place module has finished, the module voicing is called. Whether voicing or not occurs is determined by the user. The spectrogram is viewed to see if periodicity occurs in the segment under question. As in the interviews with Dr. Houde, it sometimes became necessary to expand the time axis, so individual pitch periods could be detected.

Once the call to the voicing module is completed, the main module (fric) uses the information gained in its two subroutines to determine the identity of the fricative. The identity can be specifically one of the eight fricatives, stated as a weak fricative either voiced or

unvoiced, a strong fricative either voiced or unvoiced, or no identity at all in the case where a C subroutine returns an error.

Although many improvements could be made to Fric, after the modifications described were implemented, development on the system halted and the final testing procedure began.

5. Results

5.1. Testing

The amount of testing Fric was subjected to was limited by the memory requirements of the speech data. The data files require approximately 190 Kbytes for 1 second of speech. The system on which fric was running had been operating at about 97% capacity for months, consequently few data files could be generated at one time. Testing showed this approach to be a valid method for phoneme identification and was successful in indicating shortcomings of the system.

When determining a testing technique, the space limitations presented a problem. Of the two schools of software testing, black box testing, which requires all possible inputs be tested, was not feasible. The alternative technique, white box testing, which mandates all paths in the software be tested, was chosen. Because the segmenter that will supply Fric with data has not been designed, it was unreasonable to attempt to duplicate non-fricative data that could be mistakenly sent to Fric. Therefore, no data with a low zero-crossing rate, which is considered non-fricative, was used.

The testing process involved obtaining speech data by having various speakers talk into a microphone in a relatively noise free environment. The utterances (Figure N) were well-articulated samples of continuous speech and some single words which were then processed by Speech Tool.

A total of 43 fricatives from 4 speakers, three male and one female, were identified by Fric. An identification was considered correct when Fric classified the sound as a single phoneme and the classification agreed with that of the tester.

Correct identification was made 60% of the time. Included as incorrect were cases where Fric did not have enough knowledge to decide between competing candidates and therefore gave two choices as to the identity of the phoneme. Of the identifications considered incorrect, 41% were classifications that gave correct information about the segment, but did not identify it as a single phoneme. It is reasonable to assume that with more

speaker	fricatives	noisy sounds	identified	phrase
male	2	2	2	six
male	5	5	4	three free Sunday shows
male	2	1	1	five
female	4	4	2	fathom zoophyte
female	2	2	0	six
female	4	2	1	shove biff over
female	4	4	1	thin feathers
female	2	2	0	thither
female	2	2	2	five
male	4	4	3	six of these
male	3	1	1	shove biff
male	5	5	3	sue fixed the glass
male	5	5	3	the fifth oaf is
male	4	4	3	vote then fresh

Test Phrases

Figure N

knowledge gained from continuing the interview process, more of these general classifications could be narrowed down to a specific phoneme.

Of the incorrect identifications, 41% were strong fricatives that were misclassified as weak by the C routine. There are two possible remedies to this problem. One is to lower the energy threshold required to classify a segment as strong and the other is not to overlook the implications of the spectral shape.

Since spectrograms of male voices are known to be easier to read, it was reasonable to expect Fric to function better with male voices. Indeed, in utterances by male speakers, fricatives were correctly identified 74% of the time whereas with female speakers the correct identification rate was only 37%.

Most of the errors that were made with female voices were because of a poor initial diagnosis of the amount of energy in the speech signal. Apart from the threshold problem previously mentioned, this misdiagnosis could be due to the processing of the speech signal. Speech Tool accepts energy up to 6 KHz for processing, but many female voices have ranges

up to 8 KHz. Since the strong fricatives (/s/ and /z/) have a concentration of energy in the high frequencies of the speaker's range, it is possible that the concentrations expected for strong fricatives from female speakers are being cut off by the processing of the data passed to Fric. With this amount of the speech signal missing, the sound is identified as weak. In order to remedy this problem, Speech Tool should work with frequencies up to 8KHz.

Even with this flaw, Fric competed rather well against one of the experts. Because Fric gained knowledge from two sources, it could actually identify some weak fricatives as single phonemes that one of the experts could only label as weak. From the other expert Fric acquired a method for identifying weak fricatives by tracking formant movements. Previous to this acquisition, Fric did not discriminate between the /f/ and /th/ sounds. Fric not only functions as a fricative recognizer, but also shows that this approach to phoneme recognition is well worth implementation.

5.2. RuleMaster

As RuleMaster was instrumental in the creation of Fric, it deserves discussion here. Overall it was found to be a useful tool for expert system construction. The speed with which systems could be assembled, altered and reassembled greatly increased the productivity of the time spent at the computer. The RuleMaker component of the tool, which is largely responsible for the speed, is a great enhancement to knowledge acquisition.

Some of the problems encountered while using RuleMaster were caused by the lack of support available locally. The documentation was generally good, but there were discrepancies between the actual functioning of RuleMaster and its documented powers. Having the knowledgeable users in Austin, Texas, while the system development was taking place in Rochester, New York, left no easy way to resolve problems. (See appendix 4 for description of a problem of incomplete explanation.)

Expert systems created with RuleMaster should be able to be rerun without reassembling the code. This was found to be true until the interactive sections of code were replaced with calls to C routines. The rerun stopped when attempting to recall the C

routines, because those processes were no longer running. An attempt was made to solve this problem by making the C routines infinite processes. Each routine was made into an infinite loop in which a call was made to the process which does the actual work required. This did not rectify the problem, so every time a piece of data was to be tested, Fric had to be restarted, which involved waiting while all the pre-run checks were made.

On the positive side, the intelligent editors are functional and well designed. Using sysed provides many shortcuts when setting up the hierarchy of the expert system. As the tree structure of the system is defined, the necessary files are created with the required name and the proper framework.

Once the framework is established, inded can be used to edit the induction file. Although it is somewhat difficult to edit fields with inded, creating conditions, actions and example tables is greatly simplified. When editing fields, it is much easier to use the vi editor. If the editors are being accessed through the menus, there is a default setting for which editor to use with each file type. To change from the use of one editor to another on an induction file through the menus requires constant changing of the default setting.

When this situation arises it is nice to abandon the menu system and access the RuleMaster utilities directly from the operating system. This saves the frustration of editing an induction file with vi, changing the default editor to inded and then having inded refuse to give access to the file due to a syntax error. In this situation one should leave RuleMaster and access vi and inded with direct calls from the operating system.

Systems may be assembled and run directly from the operating system or through the use of menus as well. It is very helpful to have menus available when RuleMaster is new and to be able to abandon them when they no longer offer an advantage.

Overall, RuleMaster was a good tool to use for this application. The knowledge representation was appropriate and the inference engine was flexible enough to allow various means of control. Without the use of this tool, development would have certainly taken much longer and been more difficult.

5.3. Knowledge Engineering

Perhaps the greatest learning experience involved with Fric was acting as a knowledge engineer. Being a fairly new discipline, there was not a great deal of guidance to be found. The following is a few of the guidelines acquired while functioning in this role.

The knowledge engineer should have a commitment from a person who is an expert in the domain. It is good to have an expert who is articulate, situated locally, interested in the project, convinced that computers are good tools, and able to spend the necessary time interviewing.

For each interview, a list of questions, problems to be presented and ideas to be discussed should be prepared so the time can be as productive as possible. Interviews should be situated in a place removed from phones and other distractions, to hold interruptions to a minimum.

It is also important to have the expert work with the system to pinpoint its weaknesses and witness its strengths. This helps the expert avoid feeling the time spent with the project is not worthwhile as well as providing expert feedback on the current state of the system.

Knowledge engineering can be an effective method of knowledge acquisition for expert systems when the necessary cooperation exists. It would be a worthwhile venture to continue this process for the development of a phoneme recognizer.

6. Conclusions

Fric's primary purpose, a module that functions as part of a speech recognition system, was partially realized. The problem of memory requirements was not solved and the project was moved to hardware incompatible with Fric. As far as Fric's intended purpose, it can do a reasonable job of recognizing fricative phonemes using the same techniques a spectrogram reader would use. With additional knowledge and appropriate routines to feed the decisions, Fric would be able to function better at recognizing fricatives.

Functioning as a knowledge engineer was interesting, rewarding and sometimes frustrating. It was difficult to start a project ignorant of the topic and be expected to converse with an expert in that field in a short period of time. It was gratifying to gain so much knowledge so quickly and to be able to put it to immediate use. The frustration came from dealing with people that were sometimes much too busy to be seen and not used to someone picking at their thought processes. Insight was gained into some of the problems confronting knowledge acquisition.

The understanding of the speech production mechanism and speech classification problems have given great insights into many of the innumerable problems facing an automatic speech understanding system. Recognition is complicated by many factors, some of which are obvious and many of which are clear only to the knowledgeable.

Learning about expert system building tools and RuleMaster in particular helped identify the problems involved in building an expert system. It also proved that using a tool can be a valuable aid in the development of an expert system.

6.1. Future Extensions

The following areas are suggested for further development:

(1) Completion of C routines to feed Fric the appropriate data. These would include a new routine to determine whether the amount of energy in the system should be regarded as large, a formant tracker, which could supply information about the movement of for-

mants to help determine spectral shape, and a routine that can determine if a sound is periodic in order to determine if voicing is present.

(2) a segmenter that can divide the incoming signal into manner classes. This module would accept the incoming signal as processed through Speech Tool, and determine what discrete units in that input should comprise segments. Each segment would then be passed to a recognizer specializing in that particular manner class.

(3) a recognizer for stops, nasals or vowel-like sounds. Although the manner classifications that would be used have not been chosen, each of these will need a module that can recognize phonemes within the classification.

Bibliography

(1) Allen, J. "A Perspective on Man-Machine Communication by Speech", Proceedings of the IEEE, Vol. 73, no. 11, Nov. 1985, pp. 1541-1550.

(2) Atal, B.S. "Linear Predictive Coding of Speech", *Computer Speech Processing*, Fallside, F., Woods, W. eds., pp. 81-124, Prentice Hall International, Englewood Cliffs, N.J., 1985.

(3) Baker, J.K. "Stochastic Modeling for Automatic Speech Understanding", *Speech Recognition*. Reddy, D. R. ed., pp. 521-542, Academic Press, N.Y., N.Y., 1975.

(4) Biles, J.A., "Presentation to RADC", June 24, 1986.

(5) Bradshaw, G.L., "Learning to Recognize Speech Sounds: A theory and model", Doctoral Dissertation, Carnegie-Mellon University, June 1985.

(6) Bridle, J.S. and Ralls, M.P. "An Approach to Speech Recognition using Synthesis by Rule", *Computer Speech Processing*, Fallside, F., Woods, W. eds., pp. 277-291, Prentice Hall International, Englewood Cliffs, N.J., 1985.

(7) Broad, D.J. and Shoup, J.E. "Concepts for Acoustic Phonetic Recognition", *Speech Recognition*, Reddy, D. R. ed., Academic Press, N.Y., N.Y., 1975.

(8) Brown, A.L. Jr. "Expert Systems: History, Structure and Methodology", IEEE VideoConference, Dec. 4, 1985.

(9) Buchanan, B.G. "Knowledge Based Systems: Guidelines for Problem Selection, Knowledge Acquisition and Validation", Texas Instrument's Artificial Intelligence Satellite Symposium, Nov. 13, 1985.

(10) Catford, J.C. "The Articulatory Possibilities of Man", *Manual of Phonetics*, Malmberg, B. ed., pp. 309-333, North Holland Publishing Co., Amsterdam, Holland, 1957.

(11) Chan, C. and Ng, K.W. "Separation of Fricatives from Aspirated Plosives by Means of Temporal Spectral Variation", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol ASSP-33, no. 4, Oct. 1985, pp 1130-7.

(12) Cohen, P.S. and Mercer, R.L. "The Phonological Component of a Speech Recognition System", *Speech Recognition*, Reddy, D. R. ed., pp. 275-320, Academic Press, N.Y., N.Y., 1975.

(13) Cole, R.; Rudnicky, A.; Zue, V. and Reddy, D.R. "Speech as Patterns on Paper", *Perception and Production of Fluent Speech*, Cole, R.A. ed., pp. 3-50, Lawrence Erlbaum Associates, Hillsdale, N.J., 1980.

(14) Davis, R.and Lenat, D.B. *Knowledge-Based Systems in Artificial Intelligence*, McGraw-Hill International Book Co., N.Y., N.Y., 1982.

(15) Denes, P.B. "Automatic Speech Recognition: Old and New Ideas", *Speech Recognition*. Reddy, D. R. ed., pp. 73-82, Academic Press, N.Y., N.Y., 1975.

(16) Erman, L.; Hayes-Roth, F.; Lesser, V.R. and Reddy, D. R. "The Hearsay II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty", ACM Computing Surveys, Vol 12, no. 2, June 1980, pp. 213-255.

(17) Fant, G. "Analysis and Features of Speech Processes", *Manual of Phonetics*, Malmberg, B. ed., pp. 173-273, North Holland Publishing Co., Amsterdam, Holland, 1957.

(18) Fant, G. "Distinctive features and phonetic dimension," *Speech Sounds and Features*, MIT Press, 1973.

(19) Feigenbaum, E.A. "AI: An overview of Knowledge Engineering and Expert Systems", Texas Instrument's Artificial Intelligence Satellite Symposium, Nov. 13, 1985.

(20) Feigenbaum, E.A. *The Handbook of Artificial Intelligence, Volume I*, Aaron Barr and Wm. Kaufman, Inc., Los Altos, Ca., 1981.

(21) Feigenbaum, E.A. and McCorduck, P. *The Fifth Generation*, Addison-Wesley, Reading, Ma., 1985.

(22) Francis, W.N. *The Structure of American English*, Ronald Press Co., New York, 1958.

(23) Harmon, P. and King, D. *Expert Systems Artificial Intelligence in Business*, John Wiley and Sons, N.Y., N.Y., 1985.

(24) Hart, A. "The Role of Induction in Knowledge Elicitation," Expert Systems, Vol. 2, No. 1, Jan. 1985, pp. 24-28.

(25) Hayes-Roth, F.; Waterman, D.A.; Lenat, D.B. *Building Expert Systems*, Addison-Wesley Publishing Co., Reading Ma., 1983.

(26) Hecker, M.H.L. "Speaker Recognition - An Interpretive Survey of the Literature." American Speech and Hearing Association, Washington D.C., Jan., 1971.

(27) Heffner, R-M. S. "Speech sounds in context", *General Phonetics*, pp. 183-245, The University of Wisconsin Press, 1960.

(28) Jelinek, F. "The Development of an Experimental Discrete Dictation Recognizer", Proceedings of the IEEE, Vol. 73, No. 11, Nov. 1985, pp. 1616-1624.

(29) Judson, L.; Spicer, V. and Weaver, A.T. *Voice Science*, Meredith Publishing Co., N.Y., N.Y. 1965.

(30) Klatt, D. "Linguistic uses of segmental duration in English: Acoustic and perceptual evidence", Journal Acoustic Society of America, Vol. 59, No. 5, 1976, pp. 1208-1221.

(31) Klatt, D. "Review of the ARPA Speech Understanding Project", Journal Acoustic Society of America, Vol. 62, 1976, pp.1345-1366.

(32) Klatt, D. "Scriber and Lafs: Two new Approaches to Speech Analysis", *Trends in Speech Recognition*, Lea, W.A. ed., pp. 529-55, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1980.

(33) Klatt, D. and Stevens, K.N. "On the Automatic Recognition of Continuous Speech, Implications from Spectrogram-reading experiment," IEEE Trans. Audio and Electroacoustic, Vol. AU-21, June 1973, pp. 210-217.

(34) Lea, W. "Speech Recognition: Past, Present and Future", *Trends in Speech Recognition*, Lea, W.A. ed., pp. 39-99, Prentice Hall, Inc., Englewood Cliffs, N.J., 1980.

(35) Levinson, S.E. "Structural Methods in Automatic Speech Recognition", Proceedings of the IEEE, Vol. 73, no. 11, Nov. 1985, pp. 1625-1650.

(36) Michie, D.; Muggleton, S.; Reise, C.; and Zubrick, S. "RuleMaster A Second Generation Knowledge Engineering Facility," Radian Technical Report M1-R-623, 1984.

(37) Potter, R.; Kopp, G.A. and Kopp H.G. *Visible Speech*, Dover Publications, N.Y., N.Y., 1966.

(38) Quinlan, J.R. "Semi-autonomous Acquisition of Pattern Knowledge", Machine Intelligence, pp. 159-172, Halsted Press, 1982.

(39) Radian Corporation *RuleMaster Reference Manual*, Radian Corp., Austin, Texas, Nov. 1985.

(40) Reise, C.E. and Zubrick, S.M. "RuleMaster - Using Induction to Combine Declarative and Procedural Knowledge Representations," Radian Technical Report R1-R-00299, 1985.

(41) Shoup, J.E. "Phonological Aspects of Speech Recognition", *Trends in Speech Recognition*, Lea, W.A. ed., pp. 125-138, Prentice Hall, Inc.,Englewood Cliffs, N.J. 1980.

(42) Vaissiere, J. "Speech Recognition: A Tutorial", *Computer Speech Processing*, Fallside, F., Woods, W. eds., pp.191-243, Prentice Hall International, Englewood Cliffs, N.J., 1985.

(43) Waterman, D.A. *A Guide to Expert Systems*, Addison-Wesley, Reading, Ma.,1986.

(44) Waterman, D.A. and Jenkins, B. "Heuristic Modeling Using Rule-based Computer Systems", *Terrorism: Threat, Reality, Response*, Hoover Press, 1979.

(45) Winston, P.H. *Artificial Intelligence*, Addison-Wesley, Reading, Ma., 1984.

(46) Woods, W.A. "Language Processing for Speech Understanding", *Computer Speech Processing.* Fallside, F., Woods, W. eds., pp. 305-333, Prentice Hall International, Englewood Cliffs, N.J., 1985.

(47) Zue, V. "Notes on Spectrogram Reading", Preliminary Draft, M.I.T., 1985.

(48) Zue, V. "The Use of Speech Knowledge in Automatic Speech Recognition". Proceedings of the IEEE. Vol. 73, no. 11, Nov. 1985, pp. 1602-1615.

(49) Zue, V. and Cole, R.A. "Experiments on Spectrogram Reading", ICASSP 79 Record.

(50) Zue, V.and Lamel, L.F. "An Expert Spectrogram Reader: A Knowledge-based Approach to Speech Recognition", ICASSP 1986.

(51) Zue, V. and Schwartz, R.M. "Acoustic Processing and Phonetic Analysis", *Trends in Speech Recognition,* Lea, W.A. ed., pp. 101-124, Prentice Hall, Inc., Englewood Cliffs, N.J. 1980.

Appendix 1

Preliminary System

This first system was composed of three modules: fric, place and voicing. This appendix first includes the induction file for each of these and then the corresponding Radial code.

```
/* This top module calls two child modules,
one to discern the place and the other the
manner of articulation.  When a conclusion
is reached, the identity of the fricative
is printed to output. */

MODULE: fric
DECLARATIONS:
[intent:" classify\the fricative"
     CHILD: place
     CHILD: voicing
     ]

STATE: name
   ACTIONS:
        f                [advise "could be f as in foo"]
        v                [advise "could be v as in vote"]
        T                [advise "could be T as in thin"]
        D                [advise "could be D as in then"]
        s                [advise "could be s as in see"]
        z                [advise "could be z as in zoo"]
        S                [advise "could be S as in she"]
        Z                [advise "could be Z as in azure"]

   CONDITIONS:
        place            [place] {labiodental  alveolar  palatal  dental}
        voicing          [voicing] {present  absent}

   EXAMPLES:
labiodental       absent          => (f,GOAL)
labiodental       present             => (v,GOAL)
alveolar          absent          => (s,GOAL)
alveolar          present             => (z,GOAL)
palatal           absent          => (S,GOAL)
palatal           present             => (Z,GOAL)
dental            absent          => (T,GOAL)
dental            present             => (D,GOAL)
```

```
/* This first attempt at discerning the place of
articulation assumes that all requested information
will feed into recognizing the fricative correctly.
No attempt at conflict resolution is made. */

MODULE: fric.place

DECLARATIONS:
[intent: "discern\the place of articulation\"
out: string place]

   ACTIONS:
        palatal          [prints "place is palatal \n"; "palatal" -> place]
        alveolar         [prints "place is alveolar \n"; "alveolar" -> place]
        labiodent        [prints "place is labiodental \n"; "labiodental" -> place]
        dental           [prints "place is dental \n"; "dental" -> place]

STATE: only
   CONDITIONS:
        energy           [ask "Is the energy weak or strong?" "weak,strong,can't_tell"]  ( weak  strong  can't_tell
        follow_formant   [ask "Are the following formants rising or falling?" "rising,falling"]  ( rising  falling
        mid_weakness     [ask "What is condition of energy mid_production?" "weakens,steady"]  ( weakens  steady )
        frequency        [ask "Is energy in sound low or high?" "low,high"]  ( low  high )

   EXAMPLES:
strong          -              steady          low             => (palatal,GOAL)
strong          -              steady          high            => (alveolar,GOAL)
weak            rising         -               -               => (labiodent,GOAL)
weak            falling        -               -               => (dental,GOAL)
```

```
/* Here the manner of articulation is investigated.
This is the first attempt to expand the original stub. */

MODULE: fric.voicing
DECLARATIONS:
[intent: "discern\voicing\"
out: string voicing]

STATE: only
   ACTIONS:
        yes              ["present" -> voicing]
        no               ["absent" -> voicing]

   CONDITIONS:
        voicebar         [ask "Is there a voice bar present?" "yes,no"]  { yes  no }
        longvow_shtfric  [ask "Is the fricative short, preceded by a long vowel ?" "yes,no"]  { yes  no ]
        shtvow_longfric  [ask "Is the fricative long, preceded by a short vowel?" "yes,no"]  [ yes  no }
        periodicity      [ask "Is there evidence of periodicity?" "evidence, noevidence"]  { evidence  noevidence }

   EXAMPLES:
yes             -               -               -               => (yes,GOAL)
no              yes             -               -               => (yes,GOAL)
no              no              yes             -               => (no,GOAL)
no              no              no              evidence        => (yes,GOAL)
no              no              no              noevidence      => (no,GOAL)
```

```
MODULE: fric
intent: "classify\the fricative\"
     CHILD: place
     CHILD: voicing


    STATE: name
     IF (place) IS
        "labiodental" : IF (voicing) IS
            "present" : ( advise "could be v as in vote", GOAL )
            ELSE ( advise "could be f as in foo", GOAL )
        "alveolar" : IF (voicing) IS
            "present" : ( advise "could be z as in zoo", GOAL )
            ELSE ( advise "could be s as in see", GOAL )
        "palatal" : IF (voicing) IS
            "present" : ( advise "could be Z as in azure", GOAL )
            ELSE ( advise "could be S as in she", GOAL )
        ELSE IF (voicing) IS
            "present" : ( advise "could be D as in then", GOAL )
            ELSE ( advise 'could be T as in thin", GOAL )

GOAL OF fric

   MODULE: fric.voicing
   intent: "discern\voicing\"
   out: string voicing

       STATE: only
        IF (ask "Is there a voice bar present?" "yes,no") IS
           "yes" : ( "present" -> voicing, GOAL )
           ELSE IF (ask "Is the fricative short, preceded by a long vowel ?" "yes,no") IS
               "yes" : ( "present" -> voicing, GOAL )
               ELSE IF (ask "Is the fricative long, preceded by a short vowel?" "yes,no") IS
                   "yes" : ( "absent" -> voicing, GOAL )
                   ELSE IF (ask "Is there evidence of periodicity?" "evidence, noevidence") IS
                       "evidence" : ( "present" -> voicing, GOAL )
                       ELSE ( "absent" -> voicing, GOAL )

   GOAL OF voicing

MODULE: fric.place
intent: "discern\the place of articulation\"
out: string place

   STATE: only
    IF (ask "Is the energy weak or strong?" "weak,strong,can't_tell") IS
       "weak" : IF (ask "Are the following formants rising or falling?" "rising,falling") IS
           "rising" : ( prints "place is labiodental \n"; "labiodental" -> place, GOAL )
           "falling" : ( prints "place is dental \n"; "dental" -> place, GOAL )
           ELSE ( prints "*** UNDEFINED LEAF ***\n", GOAL)
       "strong" : IF (ask "Is energy in sound low or high?" "low,high") IS
           "low" : ( prints "place is palatal \n"; "palatal" -> place, GOAL )
           ELSE ( prints "place is alveolar \n"; "alveolar" -> place, GOAL )
       ELSE ( prints "*** UNDEFINED LEAF ***\n", GOAL)

GOAL OF place
```

Appendix 2

Second Place Module

This second attempt at finding the place of articulation replaced the code shown in appendix 1.

```
/* This second attempt at determining the place of
articulation attempts to resolvc conflicting cues by
secking more information in the besure_weak and
besure_strong states. */

MODULE: fric.placc
DECLARATIONS:
[intcnt: "discern\the place of articulation\"
out: string place]

  ACTIONS:
    palatal          [prints "place is palatal \n"; "palatal" -> place]
    alvcolar         [prints "place is alveolar \n"; "alveolar" -> place]
    labiodent        [prints "place is labiodental \n"; "labiodental" -> place]
    dental           [prints "place is dcntal \n"; "dental" -> place]
    null             [null]

STATE: determine_energy
  CONDITIONS:
    energy           [ask "Is the energy weak or strong?" "weak,strong,can't_tell"]
                     { weak  strong  can't_tell }
    mid_weakness     [ask "What is condition of energy mid_production?" "weakens,steady"]
                     { weakens  steady }

  EXAMPLES:
strong          -    => (null,besure_strong)
weak            -    => (null,besure_weak)
can't_tell      -    => (null,still_unsure)

STATE: besure_weak
  ACTIONS:
    explain          [advise "Although the signal appears to be weak,
                     the presence of high frequency energy leads to
                     the conclusion the signal is classifiable strong."]

  CONDITIONS:
    conflict         [ask "Is there significant high frequency?" "significant,notsignificant"]
                     { significant  notsignificant }

  EXAMPLES:
significant     => (explain,is_strong)
notsignificant  => (null,is_weak)
```

```
STATE: besure_strong
  ACTIONS:
     explain          [advise "Although the signal appears to be strong,
                       the diffuse distribution of the energy leads to
                       the conclusion the signal is classifiably weak."]

  CONDITIONS:
     conflict         [ask "Is the energy diffuse?" "diffuse, notdiffuse"]
                      { diffuse  notdiffuse }

  EXAMPLES:
diffuse          => (explain,is_weak)
notdiffuse       => (null,is_strong)

STATE: is_strong
  CONDITIONS:
     frequency        [ask "Is energy in sound low or high?" "low,high,can't_tell"]
                       { low  high  can't_tell }

  EXAMPLES:
low              => (palatal,GOAL)
high             => (alveolar,GOAL)
can't_tell       => (null,still_unsure)

STATE: is_weak
  CONDITIONS:
     follow_formant   [ask "Are the following formants rising or falling?" "rising,falling,can't_tell"]
                      { rising  falling  can't_tell }
     prec_formant     [ask "Are the preceeding formants rising or falling?" "rising,falling,can't_tell"]
                      { rising  falling  can't_tell }

  EXAMPLES:
rising           -              => (labiodent,GOAL)
falling          -              => (dental,GOAL)
can't_tell       rising         => (dental,GOAL)
can't_tell       falling        => (labiodent,GOAL)
can't_tell       can't_tell     => (null,still_unsure)

STATE: still_unsure
  CONDITIONS:
     spec_shape       [ask "What is the spectral shape?" "high_high, low_high, diffuse"]
                      { high_high  low_high  diffuse }
     detail_specshape [ask "Is the shape flat or compact?" "flat,compact"]
                      { flat  compact }

  EXAMPLES:
high_high        -              => (alveolar,GOAL)
low_high         -              => (palatal,GOAL)
diffuse          flat           => (labiodent,GOAL)
diffuse          compact        => (dental,GOAL)
```

Appendix 3

Final Version of Interactive System

Shown in this appendix is the framework file, followed by the induction files for the modules fric, place, is_weak, is_strong and voicing. Then the Radial code corresponding to each induction file is displayed.

```
SYSTEM: fric
INDUCTION:
  fric  place  is_weak  is_strong  voicing
RADIAL:

PACKAGES:

PROMPT:
```

```
/* This top module calls two child modules,
one to discern the place and the other the
manner of articulation.  When a conclusion
is reached, the identity of the fricative
is printed to output.

With this implementation it is now possible
to get multiple identities for the fricative
if the evidence is not conclusive. */

MODULE: fric
DECLARATIONS:
[intent: "classify\the fricative\"
     CHILD: place
     CHILD: voicing
     ]


   STATE: name
   ACTIONS:
        f                [advise "could be f as in foo"]
        v                [advise "could be v as in vote"]
        T                [advise "could be T as in thin"]
        D                [advise "could be D as in then"]
        s                [advise "could be s as in see"]
        z                [advise "could be z as in zoo"]
        S                [advise "could be S as in she"]
        Z                [advise "could be Z as in azure"]
        vD               [advise "could be v as in vote or D as in then"]
        fT               [advise "could be f as in foo or T as in thin"]

   CONDITIONS:
        place            [place]  { labiodental  alveolar  palatal  dental  uncertain }
        voicing          [voicing]  { present  absent }

   EXAMPLES:
labiodental       absent            => (f,GOAL)
labiodental       present           => (v,GOAL)
alveolar          absent            => (s,GOAL)
alveolar          present           => (z,GOAL)
palatal           absent            => (S,GOAL)
palatal           present           => (Z,GOAL)
dental            absent            => (T,GOAL)
dental            present           => (D,GOAL)
uncertain         present           => (vD,GOAL)
uncertain         absent            => (fT,GOAL)
```

```
/*This is the fourth attempt at discerning the place
of articulation.  This module now attempts to classify
the sound as weak or strong and then delegates further
discrimination to the child modules.*/

MODULE: fric.place

DECLARATIONS:
[intent: "discriminate\between strong and weak sounds\,sound is"
    CHILD: is_weak
    CHILD: is_strong
    OUT: string  place]

   ACTIONS:
        is_strong          [advise "Sound is strong ";is_strong -> place]
        is_weak            [advise "Sound is weak ";is_weak -> place]

STATE: determine_energy
   CONDITIONS:
        energy             [ask "Is the amount of total energy small?"
                                "small, other"]  { small  other }
        shape              [ask "Is the spectrum flat or are there concentrations of energy?" "flat, concent"]  { flat

   EXAMPLES:
small            -                  => (is_weak,GOAL)
other            flat               => (is_weak,GOAL)
other            concent            => (is_strong,GOAL)



/*This module is reached when the sound
has enough energy that it is a strong fricative.*/

MODULE: fric.place.is_strong

DECLARATIONS:
[intent: "discriminate\between the strong sounds\"
out: string pl]

   ACTIONS:
        palatal            [advise "place is palatal ";nl; "palatal" -> pl]
        alveolar           [advise "place is alveolar ";nl; "alveolar" -> pl]
        unknown            [advise "place is unknown ";nl; "unknown" -> pl]

STATE: only
   CONDITIONS:
        bands     [ask "Is the concentration of energy in the high or mid range?"
                   "high, mid, unclear"]  { high  mid  unclear }

   EXAMPLES:
high               => (alveolar,GOAL)
mid                => (palatal,GOAL)
unclear            => (unknown,GOAL)
```

```
/*This module is reached when the signal is
weak or diffuse enough to assume it is a
weak fricative .*/

MODULE: fric.place.is_weak

DECLARATIONS:
[intent: "discriminate\between the weak sounds\"
out : string pl]

   ACTIONS:
        labiodent          [advise "place is labiodental ";nl;
                                "labiodental" -> pl]
        dental             [advise "place is dental ";nl;
                                "dental" -> pl]
        unknown            [advise "place is unknown ";nl;
                               "unknown" -> pl]

STATE: vowel_id
   CONDITIONS:
        prec_formant       [ask "What is the frequency of the second formant
preceeding this sound?" "above_1800, below_1800, not_there"]  { above_1800  below_1800  not_there }


        track_prec_form    [ask "Is the second formant falling into the sound,
rising into the sound, or steady?" "falling, rising, steady"]  { falling  rising  steady }

   EXAMPLES:
above_1800        falling            => (dental,GOAL)
above_1800        rising             => (labiodent,GOAL)
above_1800        steady             => (unknown,GOAL)
below_1800        falling            => (labiodent,GOAL)
below_1800        rising             => (dental,GOAL)
below_1800        steady             => (unknown,GOAL)
not_there         -                  => (unknown,GOAL)
```

```
/*This module attempts to discern if
voicing is evident in the sound.  When
it makes its decision it returns the
conclusion in the variable voicing.*/

MODULE: fric.voicing
DECLARATIONS:
[intent: "discern\voicing\"
out: string voicing]

STATE: only
   ACTIONS:
        yes              ["present" -> voicing]
        no               ["absent" -> voicing]

   CONDITIONS:
        voicebar         [ask "Is there a voice bar present?" "yes,no"]  { yes  no }
        periodicity      [ask "Is there evidence of periodicity?" "evidence, noevidence"]  { evidence  noevidence }

   EXAMPLES:
yes             -                =>  (yes,GOAL)
no              evidence         =>  (yes,GOAL)
no              noevidence       =>  (no,GOAL)
```

```
MODULE: fric
intent: "classify\the fricative\"
    CHILD: place
    CHILD: voicing


   STATE: name
    IF (place) IS
       "labiodental" : IF (voicing) IS
           "present" : ( advise "could be v as in vote", GOAL )
           ELSE ( advise "could be f as in foo", GOAL )
       "alveolar" : IF (voicing) IS
           "present" : ( advise "could be z as in zoo", GOAL )
           ELSE ( advise "could be s as in see", GOAL )
       "palatal" : IF (voicing) IS
           "present" : ( advise "could be Z as in azure", GOAL )
           ELSE ( advise "could be S as in she", GOAL )
       "dental" : IF (voicing) IS
           "present" : ( advise "could be D as in then", GOAL )
           ELSE ( advise "could be T as in thin", GOAL )
       ELSE IF (voicing) IS
           "present" : ( advise "could be v as in vote or D as in then", GOAL )
           ELSE ( advise "could be f as in foo or T as in thin", GOAL )

  GOAL OF fric


MODULE: fric.place
intent: "discriminate\between strong and weak sounds\,sound is"
    CHILD: is_weak
    CHILD: is_strong
    OUT: string  place

    STATE: determine_energy
     IF (ask "Is the amount of total energy small?"
                                "small, other") IS
        "small" : ( advise "Sound is weak ";is_weak -> place, GOAL )
        ELSE IF (ask "Is the spectrum flat or are there concentrations of energy?" "flat, concent") IS
            "flat" : ( advise "Sound is weak ";is_weak -> place, GOAL )
            ELSE ( advise "Sound is strong ";is_strong -> place, GOAL )

GOAL OF place

MODULE: fric.place.is_strong
intent: "discriminate\between the strong sounds\"
out: string pl

    STATE: only
     IF (ask "Is the concentration of energy in the high or mid range?"
                 "high, mid, unclear") IS
       "high" : ( advise "place is alveolar ";nl; "alveolar" -> pl, GOAL )
       "mid" : ( advise "place is palatal ";nl; "palatal" -> pl, GOAL )
       ELSE ( advise "place is unknown ";nl; "unknown" -> pl, GOAL )

GOAL OF is_strong
```

```
MODULE: fric.place.is_weak
intent: "discriminate\between the weak sounds\"
out : string pl

    STATE: vowel_id
     IF (ask "What is the frequency of the second formant
preceeding this sound?" "above_1800, below_1800, not_there") IS
       "above_1800" : IF (ask "Is the second formant falling into the sound,
rising into the sound, or steady?" "falling, rising, steady") IS
            "falling" : ( advise "place is dental ";nl;
                          "dental" -> pl, GOAL )
            "rising" : ( advise "place is labiodental ";nl;
                          "labiodental" -> pl, GOAL )
            ELSE ( advise "place is unknown ";nl;
                         "unknown" -> pl, GOAL )
       "below_1800" : IF (ask "Is the second formant falling into the sound,
rising into the sound, or steady?" "falling, rising, steady") IS
            "falling" : ( advise "place is labiodental ";nl;
                          "labiodental" -> pl, GOAL )
            "rising" : ( advise "place is dental ";nl;
                          "dental" -> pl, GOAL )
            ELSE ( advise "place is unknown ";nl;
                         "unknown" -> pl, GOAL )
       ELSE ( advise "place is unknown ";nl;
                         "unknown" -> pl, GOAL )

GOAL OF is_weak
```

```
MODULE: fric.voicing
intent: "discern\voicing\"
out: string voicing

    STATE: only
     IF (ask "Is there a voice bar present?" "yes,no") IS
        "yes" : ( "present" -> voicing, GOAL )
        ELSE IF (ask "Is there evidence of periodicity?" "evidence, noevidence") IS
            "evidence" : ( "present" -> voicing, GOAL )
            ELSE ( "absent" -> voicing, GOAL )

GOAL OF voicing
```

Appendix 4

Explanation Problem

In the following script the explanation given for determination of the place of articulation is not sufficient. The prompt file invoked in fric.place gives an explanation that only tells which alternative of the menu was chosen, but not what the content of that choice was. This would make it extremely difficult for anyone who did not have a great memory to understand the explanation.

The second part of determining which strong fricative was spoken is related by the ask2 question about concentrations in fric.place.is_strong. This determination is totally absent from the explanation although it is crucial information without which the discrimination between the s and sh sounds would be impossible.

Apr 13 15:07:44 1986

sun% run fric

Radial Version UX-00.15
Co-developed By Intelligent Terminals Limited And Radian Corporation
Property Of Intelligent Terminals Limited - Confidential Licensed Materials
Exclusively Licensed In North America To Radian Corporation
Parsing Radial program from <fric.run>

Attaching

Executing module <fric>

Does the sound segment have

a) concentrations of energy mid or high range

b) very diffuse energy

c) too little energy to tell
> a

Is there
a) a concentration of energy about 5kHz
b) a concentration of energy about 3kHz
c) both of these a) and b)
d) neither of these [a,b,c,d] why

****

An investigation of the energy concentration is being performed
in order to discriminate between the s (alveolar) and sh (palatal) sounds

At <fric>
c)ont e)lab h)elp : c

****

Is there
a) a concentration of energy about 5kHz
b) a concentration of energy about 3kHz
c) both of these a) and b)
d) neither of these [a,b,c,d] a
place is alveolar
Is there a voice bar present? [yes,no] n
Is there evidence of periodicity? [evidence, noevidence] n

Advice: could be s as in see

(RETURN continues) why

****

Since the answer to 'Is there evidence of periodicity?' is noevidence
when the answer to 'Is there a voice bar present?' is no
it follows that voicing is absent
Since the reply to 'strong' is a
it follows that the place of articulation is alveolar
Since voicing is absent
when the place of articulation is alveolar
it is necessary to open the prompt file 'prompter.prb'
and advise 'could be s as in see'
in order to classify the fricative

At <fric>
c)ont e)lab h)elp : c

****

(RETURN continues)

At <fric>
a)ttch c)hld e)xpln h)elp q)uit r)un tr)ee : r

Executing module <fric>

Does the sound segment have

a) concentrations of energy mid or high range

b) very diffuse energy

c) too little energy to tell
>da^H
Did not understand your response.
Please try again

Does the sound segment have

a) concentrations of energy mid or high range

b) very diffuse energy

c) too little energy to tell
> ?
If you can't tell, looking at the waveform
can sometimes let you know about the amount
of energy.>
Did not understand your response.
Please try again

Does the sound segment have

a) concentrations of energy mid or high range

b) very diffuse energy

c) too little energy to tell
> a
Is there
a) a concentration of energy about 5kHz
b) a concentration of energy about 3kHz
c) both of these a) and b)
d) neither of these [a,b,c,d] b
place is palatal
Is there a voice bar present? [yes,no] y

Advice: could be Z as in azure

(RETURN continues) why

****
Since the answer to 'Is there a voice bar present?' is yes
it follows that voicing is present
Since the reply to 'strong' is a
when the reply to 'strong' is d
and the reply to 'strong' is d
it follows that the place of articulation is palatal
Since voicing is present
when the place of articulation is palatal
it is necessary to open the prompt file 'prompter.prb'

```
   c)ont e)lab h)elp : e

*********************************************************************************

Advice: could be Z as in azure

(RETURN continues)

At <fric>
a)ttch c)hld e)xpln h)elp q)uit r)un tr)ee : r

Executing module <fric>


Does the sound segment have

   a) concentrations of energy mid or high range

   b) very diffuse energy

   c) too little energy to tell
     > a
Is there
  a) a concentration of energy about 5kHz
  b) a concentration of energy about 3kHz
  c) both of these a) and b)
  d) neither of these  [a,b,c,d] c
place is palatal
Is there a voice bar present? [yes,no] n
Is there evidence of periodicity? [evidence,n noevidence]

Advice: could be S as in she

(RETURN continues) why

*********************************************************************************
Since the answer to 'Is there evidence of periodicity?' is noevidence
    when the answer to 'Is there a voice bar present?' is no
    it follows that voicing is absent
Since the reply to 'strong' is a
    it follows that the place of articulation is palatal
Since voicing is absent
    when the place of articulation is palatal
    it is necessary to open the prompt file 'prompter.prb'
        and advise 'could be S as in she'
    in order to classify the fricative

    At <fric>
    c)ont e)lab h)elp : c

*********************************************************************************

Advice: could be S as in she

(RETURN continues)

At <fric>
a)ttch c)hld e)xpln h)elp q)uit r)un tr)ee : r

Executing module <fric>


Does the sound segment have
```

```
   b) very diffuse energy

   c) too little energy to tell
     > a
Is there
  a) a concentration of energy about 5kHz
  b) a concentration of energy about 3kHz
  c) both of these a) and b)
  d) neither of these  [a,b,c,d] why

*******************************************************************************
An investigation of the energy concentration is being performed
    in order to discriminate between the s (alveolar) and sh (palatal) sounds

    At <fric>
    c)ont e)lab h)elp : c

*******************************************************************************
Is there
  a) a concentration of energy about 5kHz
  b) a concentration of energy about 3kHz
  c) both of these a) and b)
  d) neither of these  [a,b,c,d] ?

must be prefix of one of [a,b,c,d]


*** Press RETURN to continue
Is there
  a) a concentration of energy about 5kHz
  b) a concentration of energy about 3kHz
  c) both of these a) and b)
  d) neither of these  [a,b,c,d] a
place is alveolar
Is there a voice bar present? [yes,no] why

*******************************************************************************
An investigation of the answer to 'Is there a voice bar present?' is being performed
    in order to discern voicing

    At <fric>
    c)ont e)lab h)elp : c

*******************************************************************************
Is there a voice bar present? [yes,no] n
Is there evidence of periodicity? [evidence, noevidence] n

Advice: could be s as in see

(RETURN continues)

At <fric>
a)ttch c)hld e)xpln h)elp q)uit r)un tr)ee : e

*******************************************************************************
    Since the answer to 'Is there evidence of periodicity?' is noevidence
        when the answer to 'Is there a voice bar present?' is no
        it follows that voicing is absent
    Since the reply to 'strong' is a
        it follows that the place of articulation is alveolar
    Since voicing is absent
        when the place of articulation is alveolar
        it is necessary to open the prompt file 'prompter.prb'
            and advise 'could be s as in see'
        in order to classify the fricative
```

Appendix 5

Final System

The final version of Fric is shown here as Radial code. This is the fric.run file that is created during the assembly phase. Some of the primitive modules are shown here, but others, which are not called by this program have been omitted.

```
MODULE: main

        INTENT: "do the main problem"

        CHILD:  # (110, 1, 1)

        STATE:  only
                ( fric, GOAL)
GOAL OF main

MODULE: fric
INTENT: "classify\the fricative\"
     CHILD: place
     CHILD: voicing
     CHILD: getstuff
 LOCAL:  string filename
         integer (sframe eframe)


    STATE: get_segment
    (
    getstuff,name
)

    STATE: name
     IF (place) IS
        "labiodental" : IF (voicing) IS
            "present" : ( advise "could be v as in vote", GOAL )
            ELSE ( advise "could be f as in foo", GOAL )
        "alveolar" : IF (voicing) IS
            "present" : ( advise "could be z as in zoo", GOAL )
            ELSE ( advise "could be s as in see", GOAL )
        "palatal" : IF (voicing) IS
            "present" : ( advise "could be Z as in azure", GOAL )
            ELSE ( advise "could be S as in she", GOAL )
        "dental" : IF (voicing) IS
            "present" : ( advise "could be D as in then", GOAL )
            ELSE ( advise "could be T as in thin", GOAL )
        "weak" : IF (voicing) IS
            "present" : ( advise "could be v as in vote or D as in then", GOAL )
            ELSE ( advise "could be f as in foo or T as in thin", GOAL )
        "strong" : IF (voicing) IS
            "present" : ( advise "could be z as in zoo or Z as in azure", GOAL )
            ELSE ( advise "could be s as in see or S as in she", GOAL )
       ELSE ( advise "error in total_energy", GOAL )

GOAL OF fric

MODULE: fric.place
INTENT: " identity of the sound'"
    CHILD: is_weak
    CHILD: is_strong
 OUT: string  place

   STATE: determine_energy
```

```
       IF total_energy fric.filename fric.sframe fric.eframe IS
          "0": (advise "total_energy returned a 0 meaning small amount of energy";nl;is_weak -> place,GOAL)
          "-1": (advise "error in total_energy";nl;"garbage" -> place,GOAL)
       ELSE (advise "total_energy returned a 1 meaning large amount of energy";nl,determine_shape)


   STATE: determine_shape
    IF (ask2 "Is the spectrum flat or are there concentrations of energy?" "flat, concent" "\shape\" "flat, concentrated ir
IS
       "flat" : ( advise "Sound is weak ";nl;is_weak -> place, GOAL )
       ELSE ( advise "Sound is strong ";nl;is_strong -> place, GOAL )

GOAL OF place

MODULE: fric.place.is_weak
intent: "discriminate\between the weak sounds\"
out : string pl

   STATE: vowel_id
    IF (ask "What is the frequency of the second formant
preceeding this sound?" "above_1800, below_1800, not_there") IS
       "above_1800" : IF (ask "Is the second formant falling into the sound,
rising into the sound, or steady?" "falling, rising, steady") IS
          "falling" : ( advise "place is dental (th sound)";nl;
"dental" -> pl, GOAL )
          "rising" : ( advise "place is labiodental (f sound)";nl;
"labiodental" -> pl, GOAL )
          ELSE ( null, vowel_id2 )
       "below_1800" : IF (ask "Is the second formant falling into the sound,
rising into the sound, or steady?" "falling, rising, steady") IS
          "falling" : ( advise "place is labiodental (f sound)";nl;
"labiodental" -> pl, GOAL )
          "rising" : ( advise "place is dental (th sound)";nl;
"dental" -> pl, GOAL )
          ELSE ( null, vowel_id2 )
       ELSE ( null, vowel_id2 )

   STATE: vowel_id2
    IF (ask "What is the frequency of the second formant
following this segment ?" "above_1800, below_1800, not_there") IS
       "above_1800" : IF (ask "Is the second formant following the segment
rising falling or steady ?" "rising, falling steady ") IS
          "rising" : ( advise "place is dental (th sound)";nl;
"dental" -> pl, GOAL )
          "falling" : ( advise "place is labiodental (f sound)";nl;
"labiodental" -> pl, GOAL )
          ELSE ( advise "place is weak ";nl;
"weak" -> pl, GOAL )
       "below_1800" : IF (ask "Is the second formant following the segment
rising falling or steady ?" "rising, falling steady ") IS
          "rising" : ( advise "place is labiodental (f sound)";nl;
"labiodental" -> pl, GOAL )
          "falling" : ( advise "place is dental (th sound)";nl;
"dental" -> pl, GOAL )
          ELSE ( advise "place is weak ";nl;
"weak" -> pl, GOAL )
       ELSE ( advise "place is weak ";nl;
```

```
"weak" -> pl, GOAL )

GOAL OF is_weak

MODULE: fric.place.is_strong
INTENT: "discriminate\between the strong sounds\"
    CHILD: get_ranges
 OUT:      string pl
 LOCAL:   integer (midcutoff highcutoff)


   STATE: determine_ranges
   (
      get_ranges, next
)

   STATE: next

      IF bands fric.filename fric.sframe fric.eframe midcutoff highcutoff IS
          "0": (advise "Concentrations of energy are unclear.";nl;"strong" -> pl,GOAL)
          "1": (advise "Concentrations of energy exist midrange.";nl;"palatal" -> pl,GOAL)
          "-1": (advise "error in bands";nl;"strong" -> pl,GOAL)
      ELSE (advise "Concentrations of energy are high.";nl;"alveolar" -> pl,GOAL)


GOAL OF is_strong

MODULE: fric.place.is_strong.get_ranges
MUTE:

   STATE: get_ranges
   (
      prints "Bounds should be given in bins."; nl;
      prints "Each bin corresponds to 100 Hz.";nl;
      read "What do you want as mid range lower bound?  " -> is_strong.midcutoff;
      read "What do you want as high range lower bound?  " -> is_strong.highcutoff,
      GOAL
)

GOAL OF get_ranges

MODULE: fric.voicing
intent: "discern\voicing\"
out: string voicing

   STATE: only
    IF (ask "Is there a voice bar present?" "yes,no") IS
      "yes" : ( "present" -> voicing, GOAL )
      ELSE IF (ask "Is there evidence of periodicity?" "evidence, noevidence") IS
          "evidence" : ( "present" -> voicing, GOAL )
          ELSE ( "absent" -> voicing, GOAL )

GOAL OF voicing

MODULE: fric.getstuff
MUTE: "\which segment is to be identified\"
```

```
    STATE: get_segment
    (
reads "What is the name of the file containing the utterance?  " -> fric.filename;
read "At what frame does the segment start?  " -> fric.sframe;
read "At what frame does the segment end?  " -> fric.eframe, GOAL
)

GOAL OF getstuff

PRIMITIVE MODULE: total_energy
IN:     string filename
           integer ( sframe eframe )
   OUT:    integer code

   /* code 0 is small
      code 1 is large
      code -1 is error   */

GOAL OF total_energy


PRIMITIVE MODULE: bands
IN:     string filename
           integer (sframe eframe mid high)
   OUT:    integer code

     /* code 0 is other
        code 1 is mid
        code 2 is high
        code -1 is error  */

   GOAL OF bands


PRIMITIVE GENERIC STORAGE MODULE boolean IS
       CHILD:  print   (30,0,1),
               read,
               not     (6,0,1),
               and     (5,1,1),
               xor     (4,1,1),
               or      (3,1,1)

GOAL OF boolean

MODULE boolean.print IS
       MUTE:
       IN: boolean a

       STATE: output
              IF a IS
                     "T": (prints "TRUE", GOAL)
                     "F": (prints "FALSE", GOAL)
                     ELSE (prints "boolean.print error\n", GOAL)
GOAL OF print
```

```
MODULE boolean.read IS
        INTENT: "request\the response to '$1'\"
        IN:     string message
        OUT:    boolean result

        LOCAL:  string buf

        STATE: input
                (reads message -> buf, test)

        STATE: test
                IF buf IS
                "T":    ("T" -> result, GOAL)
                "F":    ("F" -> result, GOAL)
                ELSE    (prints "Enter T or F\n", input)
GOAL OF read

PRIMITIVE MODULE boolean.not IS
        INTENT: "not $1"
        IN: boolean a
        OUT: boolean result
GOAL OF not

PRIMITIVE MODULE boolean.and IS
        INTENT: "$1 and $2"
        IN: boolean (a, b)
        OUT: boolean result
GOAL OF and

PRIMITIVE MODULE boolean.xor IS
        INTENT: "$1 xor $2"
        IN: boolean (a, b)
        OUT: boolean result
GOAL OF xor

PRIMITIVE MODULE boolean.or IS
        INTENT: "$1 or $2"
        IN: boolean (a, b)
        OUT: boolean result
GOAL OF or

PRIMITIVE GENERIC STORAGE MODULE float IS
        child:  + (20,1,1),
                - (20,1,1),
                * (30,1,1),
                ** (50,1,1),
                / (30,1,1),
                < (15,1,1),
                <= (15,1,1),
                > (15,1,1),
                >= (15,1,1),
                == (10,1,1),
                != (10,1,1),
                f_to_s (10.0,1),
                s_to_f (10.0,1),
```

```
                sqrt,
                int,
                read,
                read2,
                print {8,0,1}
GOAL OF float

PRIMITIVE MODULE float.print IS
        MUTE:
        IN:  float val
GOAL OF print

PRIMITIVE MODULE float.+ IS
        INTENT:  "$1 + $2"
        IN:  float {a,b}
        OUT: float val_out
GOAL OF +

PRIMITIVE MODULE float.* IS
        INTENT:  "$1 * $2"
        IN:  float {a,b}
        OUT: float val_out
GOAL OF *

PRIMITIVE MODULE float.** IS
        INTENT:  "$1 ** $2"
        IN:     float {a,b}
        OUT:    float result
GOAL OF **

PRIMITIVE MODULE float.f_to_s IS
        INTENT: "\the string equivalent of $1\"
        IN:     float a
        OUT:    string result
GOAL OF f_to_s

PRIMITIVE MODULE float.s_to_f IS
        INTENT: "\the float equivalent of $1\"
        IN:     string a
        OUT:    float result
GOAL OF s_to_f

PRIMITIVE MODULE float.sqrt IS
        INTENT:  "the square root of $1"
        IN: float a
        OUT: float result
GOAL OF sqrt

PRIMITIVE MODULE float.read IS
        INTENT: "request\the response to '$1'\"
        IN:     string prompt
        OUT:    float val_out
GOAL OF read

MODULE float.read2 IS
        SILENT: "$2"
```

```
        IN:     string (prompt,intention)
        OUT:    float val_out

        STATE:  only
                ( read prompt -> val_out, GOAL)

GOAL OF read2

PRIMITIVE MODULE float./ IS
        INTENT:  "$1 / $2"
        IN:  float (a,b)
        OUT: float val_out
GOAL OF /

PRIMITIVE MODULE float.- IS
        INTENT:  "$1 - $2"
        IN:  float (a,b)
        OUT: float val_out
GOAL OF -

PRIMITIVE MODULE float.< IS
        INTENT:  "$1 < $2"
        IN: float (a,b)
        OUT: boolean result
GOAL OF <

PRIMITIVE MODULE float.<= IS
        INTENT:  "$1 <= $2"
        IN: float (a,b)
        OUT: boolean result
GOAL OF <=

PRIMITIVE MODULE float.> IS
        INTENT:  "$1 > $2"
        IN: float (a,b)
        OUT: boolean result
GOAL OF >

PRIMITIVE MODULE float.>= IS
        INTENT:  "$1 >= $2"
        IN: float (a,b)
        OUT: boolean result
GOAL OF >=

PRIMITIVE MODULE float.== IS
        INTENT:  "$1 == $2"
        IN: float (a,b)
        OUT: boolean result
GOAL OF ==

PRIMITIVE MODULE float.!= IS
        INTENT:  "$1 != $2"
        IN: float (a,b)
        OUT: boolean result
GOAL OF !=
```

Appendix 6

C Routines

The code for the two C routines written by Eric Luce is found here. The first procedure, summit, is used to determine the amount of total energy found in a particular segment. If the average of the total energy per millisecond is less than or equal to 200 then the segment is considered to have a small amount of energy.

The second procedure, inquire, is used to determine where the concentrations of energy lie in a strong fricative. Two parameters are accepted in units of bins (100 Hz) and a decision is made as to whether there is more energy between the two measurements or above the upper measurement.

```
#define ERROR          -1
#define FALSE           0
#define LOW             0
#define OTHER           1
#define TRUE            1
#define TWOBYTES        2

#include <stdio.h>
#include <sys/file.h>

main()
   {
        char radial_name[50], filename[100];
        int sframe, eframe;

        scanf ("%s %s %d %d", radial_name, filename, &sframe, &eframe);
        summit (filename, sframe, eframe);
   }

summit (filename, sframe, eframe)
   char *filename;
   int sframe, eframe;

   {
        int
           fd,                  /* file descriptor of input file */
           bytes_skipped,       /* number of bytes before the sframe data */
           n,                   /* number of bytes successfully read */
           frame,               /* current frame being examined */
           done;                /* boolean indicating error status */
        register int
           x;                   /* result of calculations on input file */
        short
           buf[3000];           /* buffer to hold the number read */

        /* tack the extension on to the end of the filename */
        strcat (filename, ".sum");

        /* try to open the input file */
        if ((fd = open (filename, O_RDONLY)) == -1)
           {
                /* return to radian module with a -1 */
                printf ("%d", ERROR);
                printf ("%c", 0x04);
           }
        else
           {
                /* skip input data until starting frame */
                bytes_skipped = (TWOBYTES * sframe);
                n = read (fd, &buf[0], bytes_skipped);
                if (n != bytes_skipped)
                   {
                        /* return to radian module with a -1 */
                        printf ("%d", ERROR);
                        printf ("%c", 0x04);
                   }
                else
```

```
                {
                    done = FALSE;
                    frame = sframe;
                    x = 0;
                    while (frame <= eframe && done == FALSE)
                       {
                            n = read (fd, &buf[0], TWOBYTES);
                            if (n != TWOBYTES)
                               {
                                    /* return to radian module with a -1 */
                                    printf ("%d", ERROR);
                                    printf ("%c", 0x04);

                                    /* set done to true */
                                    done = TRUE;
                               }
                            else
                                    x += buf[0];
                            ++frame;
                       }

                    if (done == FALSE)
                       {
                            x /= (eframe - sframe + 1);
                            if (x <= 200)
                                    printf ("%d", LOW);
                            else
                                    printf ("%d", OTHER);
                            printf ("%c", 0x04);
                       }
                }
        }
}
```

```
#define         ERROR           -2
#define         FALSE            0
#define         TRUE             1
#define         TWOBYTES         2

#include <stdio.h>
#include <sys/file.h>

main ()
   {
        char radial_name[50], filename[50];
        int sframe, eframe, f1, f2;

        scanf ("%s %s %d %d %d %d", radial_name, filename, &sframe, &eframe, &f1, &f2);
        inquire (filename, sframe, eframe, f1, f2);
   }

inquire (filename, sframe, eframe, f1, f2)
   char filename[50];
   int sframe, eframe, f1, f2;

   {
        short
           dbase[3000][64];
        int
           done,
           fd,
           n,
           bytes_skipped,
           i,
           j,
           buf,
           sum,
           high,
           mid;
        float
           frac;

        /* set done to FALSE */
        done = FALSE;

        /* check for valid f1 and f2 */
        if (f1 < 0 || f2 < 0 || f1 > 63 || f2 > 63)
                done = get_out ();

        /* open the input file */
        strcat (filename, ".f");
        if ((fd = open (filename, O_RDONLY)) == -1)
                done = get_out ();
```

```
      /* if no errors have been found, process the data */
      if (done == FALSE)
         {
              i = sum = high = mid = 0;
              while ((n = read (fd, &dbase[i][0], 128)) == 128 && i <= eframe)
                 {
                      if (i >= sframe && i <= eframe)
                         {
                              for (j = 0; j <= 63; j++)
                                 {
                                      sum += dbase[i][j];
                                      if (j >= f1 && j <= f2)
                                              mid += dbase[i][j];
                                      if (j >= f2)
                                              high += dbase[i][j];
                                 }
                         }
                        ++i;
                 }
              frac = high - mid;
              frac /= sum;
              if (fabs(frac) < .1)
                      printf ("%d", 0);
              else if (high > mid)
                      printf ("%d", 2);
              else
                      printf ("%d", 1);
              printf ("%c", 0x04);
         }
        close (fd);
   }



get_out ()

   {
        printf ("%d", ERROR);
        printf ("%c", 0x04);
        return (TRUE);
   }
```